

# **[Differential Privacy in Reinforcement Learning]**

by **[Sheng Shen]**

Thesis submitted in fulfilment of the requirements for  
the degree of

**[C02029: Doctor of Philosophy]**

under the supervision of [Prof. Tianqing Zhu & Dr. Bo Liu]

University of Technology Sydney  
Faculty of [Engineering & IT]

[22-Dec-2022]

# Certificate of Original Authorship Template

Graduate research students are required to make a declaration of original authorship when they submit the thesis for examination and in the final bound copies. Please note, the Research Training Program (RTP) statement is for all students. The Certificate of Original Authorship must be placed within the thesis, immediately after the thesis title page.

## Required wording for the certificate of original authorship

### CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Sheng Shen*, declare that this thesis is submitted in fulfilment of the requirements for the award of *C02029: Doctor of Philosophy*, in the *School of Computer Science, Faculty of Engineering & IT* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

*\*If applicable, the above statement must be replaced with the collaborative doctoral degree statement (see below).*

*\*If applicable, the Indigenous Cultural and Intellectual Property (ICIP) statement must be added (see below).*

This research is supported by the Australian Government Research Training Program.

Production Note:

Signature: Signature removed prior to publication.

Date: 22/12/2022

## Collaborative doctoral research degree statement

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with *[insert collaborative partner institution]*.

## Indigenous Cultural and Intellectual Property (ICIP) statement

This thesis includes Indigenous Cultural and Intellectual Property (ICIP) belonging to *[insert relevant language, tribal or nation group(s) or communities]*, custodians or traditional owners. Where I have used ICIP, I have followed the relevant protocols and consulted with appropriate Indigenous people/communities about its inclusion in my thesis. ICIP rights are Indigenous heritage and will always remain with these groups. To use, adapt or reference the ICIP contained in this work, you will need to consult with the relevant Indigenous groups and follow cultural protocols.



*C02029: Doctor of Philosophy*  
*CRICOS Code: 00099F*  
*PhD Thesis: Computer Science*  
*December 2022*

# *Differential Privacy in Reinforcement Learning*

---

*Sheng Shen*

School of Computer Science  
Faculty of Eng. & IT  
University of Technology Sydney  
NSW - 2007, Australia



---

---

# Differential Privacy in Reinforcement Learning

---

---

*A thesis submitted in fulfilment of the requirements  
for the degree of*

Doctor of Philosophy  
*in*  
Computer Science

*by*

Sheng Shen

*to*

Center for Cyber Security and Privacy  
School of Computer Science  
Faculty of Engineering and Information Technology  
University of Technology Sydney  
NSW - 2007, Australia

December 2022







## ABSTRACT

**R**einforcement learning is a principled AI framework for autonomously experience-driven learning. The primary goal of reinforcement learning is to train autonomous agents to learn the optimal behaviors for their interactive environments. Deep reinforcement learning promotes a higher-level understanding of the visual world in the field of reinforcement learning by combining deep learning models and reinforcement learning algorithms. Since reinforcement learning is achieving great success in an increasing number of application fields that may involve huge amounts of private information, the security of policies and privacy preservation in reinforcement learning have given rise to widespread concerns. In addition, deep reinforcement learning policies parameterized by neural networks have been demonstrated to be vulnerable to adversarial attacks in supervised learning settings. Privacy leakage also occurs in multi-agent reinforcement learning systems where agents' actions or behaviors are directly exposed to other agents.

To address these multiple privacy concerns in reinforcement learning, we apply differential privacy in variant scenarios of reinforcement learning. In this thesis, we introduce our differentially private methods in those diverse scenarios to preserve privacy, including the multi-agent advising framework, multi-agent planning framework, the deep reinforcement learning context, machine learning classifiers and multi-agent game theoretic framework, respectively. We have provided detailed theoretical analysis and comprehensive experimental results to demonstrate that our methods can guarantee privacy preservation as well as the utility of reinforcement learning in diverse scenario in different chapters.



## DEDICATION

*Dedicated to my love, Maiging, how bravely tolerated all my stubbornness, temper and craziness when things did not go as expected. I am so grateful for your understanding of my choice of research, and your constant love and support in my life. You are always my spiritual support.  
I love you.*



## ACKNOWLEDGMENTS

I have received much support and assistance throughout the writing of this thesis. I would first like to acknowledge my principle supervisor Prof. Tianqing Zhu who provided much helpful and insightful advice at any time required. My fantastic journey of research began with your trust in my potential and the opportunities you offered. I am grateful to Prof. Wanlei Zhou who funds my scholarship during my PhD. Also thank Dr. Bo Liu to share with me your professional knowledge and research experience. Next, I wish to extend my thanks to Dr. Dayong Ye who directed, guided and co-authored with me on many works. I also acknowledge my colleagues at the Center for Cyber Security and Privacy, UTS, who showed enthusiasm for my research and contributed their talent in the teamwork.

Last but not least, thanks to my partner and mother who endured this long journey with me, always offering support and love. Thank you for always being my strongest support and offering me the courage to move forward.



## LIST OF PUBLICATIONS

1. Sheng SHEN, Tianqing ZHU, Dayong YE, Mengmeng YANG, Tingting LIAO & Wanlei ZHOU. (2019, December). Simultaneously advising via differential privacy in cloud servers environment. In *International Conference on Algorithms and Architectures for Parallel Processing* (pp. 550-563). Springer, Cham.
2. Sheng, SHEN, Tianqing ZHU, Dayong YE, Minghao WANG, Xuhan ZUO & Andi ZHOU. (2022). A novel differentially private advising framework in cloud server environment. *Concurrency and Computation: Practice and Experience*, 34(7), e5932.
3. Dayong YE, Tianqing ZHU, Sheng, SHEN, Wanlei ZHOU & Philip YU (2020). (2020). Differentially private multi-agent planning for logistic-like problems. *IEEE Transactions on Dependable and Secure Computing*.
4. Sheng SHEN, Dayong YE, Tianqing ZHU, & Wanlei ZHOU. (2022). Privacy Preservation in Deep Reinforcement Learning: a Training Perspective. Submitted to *IEEE Transactions on Cybernetics*.
5. Dayong YE, Sheng, SHEN, Tianqing ZHU, Bo LIU & Wanlei ZHOU. (2022). One Parameter Defense-Defending Against Data Inference Attacks via Differential Privacy. *IEEE Transactions on Information Forensics and Security*, 17, 1466-1480.
6. Dayong YE, Tianqing ZHU, Sheng SHEN, & Wanlei ZHOU. (2020). A differentially private game theoretic approach for deceiving cyber adversaries. *IEEE Transactions on Information Forensics and Security*, 16, 569-584.
7. Sheng SHEN, Tianqing ZHU, Di WU, Wei WANG, & Wanlei ZHOU. (2020). From distributed machine learning to federated learning: In the view of data privacy and security. *Concurrency and Computation: Practice and Experience*.
8. Xin CHEN, Tao ZHANG, Sheng SHEN, Tianqing ZHU, & Ping XIONG. (2021). An optimized differential privacy scheme with reinforcement learning in VANET. *Computers & Security*, 110, 102446.

- 
9. Yua0 WANG, Tianqing ZHU, Wenhan Chang, Sheng SHEN, & Wei REN. (2020, November). Model Poisoning Defense on Federated Learning: A Validation Based Approach. In *International Conference on Network and System Security* (pp. 207-223). Springer, Cham.



# TABLE OF CONTENTS

<b>List of Publications</b>	<b>ix</b>
<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Privacy Concerns . . . . .	2
1.3 Research Objective . . . . .	3
1.4 Main Contributions . . . . .	4
1.4.1 Differentially Private Advising Framework in Simultaneously Reinforcement Learning . . . . .	4
1.4.2 Differentially Private Reinforcement Learning Based Multi-agent Planning . . . . .	5
1.4.3 Privacy Preservation in Deep Reinforcement Learning: a Training Perspective . . . . .	5
1.4.4 One Parameter Defense - Defending against Data Inference Attacks via Differential Privacy . . . . .	6
1.5 Thesis Organization . . . . .	6
<b>2 Preliminaries of Reinforcement Learning and Differential Privacy</b>	<b>9</b>
2.1 Notations . . . . .	9
2.2 Reinforcement Learning . . . . .	9
2.2.1 Traditional Reinforcement Learning . . . . .	11
2.2.2 Deep Reinforcement Learning . . . . .	12
2.3 Differential Privacy . . . . .	13

<b>3</b>	<b>Differentially Private Advising Framework In Simultaneously Reinforcement Learning</b>	<b>17</b>
3.1	Introduction . . . . .	17
3.2	Problem Statement . . . . .	20
3.2.1	Scenario setting . . . . .	20
3.2.2	Scenario discussion . . . . .	21
3.3	Related Work . . . . .	22
3.4	Preliminary . . . . .	23
3.5	Differentially Private Advising Framework . . . . .	23
3.5.1	Reinforcement Learning And Normalization . . . . .	24
3.5.2	When To Ask For And Give Advice . . . . .	25
3.5.3	Algorithm Of Differentially Private Advising Framework . . . . .	28
3.6	Experiments . . . . .	31
3.6.1	Experiment Setting . . . . .	31
3.6.2	Experiment Results . . . . .	32
3.6.3	Result Discussion . . . . .	35
3.7	Summary and Future Work . . . . .	36
<b>4</b>	<b>Differentially Private Multi-Agent Planning for Logistic-like Problems</b>	<b>37</b>
4.1	Introduction . . . . .	38
4.2	Related Work . . . . .	40
4.2.1	Weak privacy-preserving approaches . . . . .	40
4.2.2	Strong privacy-preserving approaches . . . . .	40
4.2.3	Other privacy-preserving approaches . . . . .	41
4.3	A Motivating Example . . . . .	42
4.4	Preliminaries . . . . .	44
4.4.1	The planning model . . . . .	44
4.4.2	Privacy-preserving multi-agent planning . . . . .	46
4.5	The strong privacy-preserving planning approach . . . . .	48
4.5.1	Step 1: Creating a high-level map . . . . .	49
4.5.2	Step 2: Each intermediate agent provides map and route information	49
4.5.3	Step 3: Creating a complete plan . . . . .	55
4.5.4	A simplification of the proposed approach . . . . .	57
4.6	Theoretical analysis . . . . .	57
4.6.1	Soundness analysis . . . . .	57

4.6.2	Completeness analysis . . . . .	57
4.6.3	Privacy-preserving analysis . . . . .	58
4.6.4	Communication analysis . . . . .	61
4.7	Application of our approach to other domains . . . . .	63
4.7.1	Packet routing in networks . . . . .	63
4.7.2	Airplane transport . . . . .	63
4.7.3	Rover exploration . . . . .	64
4.8	Experiments . . . . .	64
4.8.1	Experimental setup . . . . .	64
4.8.2	Experimental results . . . . .	66
4.8.3	Summary . . . . .	70
4.9	Summary and Future Work . . . . .	71
<b>5</b>	<b>Privacy Preservation in Deep Reinforcement Learning: a Training Perspective</b>	<b>73</b>
5.1	Introduction . . . . .	74
5.2	Related Work . . . . .	76
5.2.1	Sensitivity of RL Training Environment . . . . .	76
5.2.2	Membership Inference Attack . . . . .	76
5.2.3	Inverse Reinforcement Learning . . . . .	77
5.2.4	Privacy Preservation in Reinforcement Learning . . . . .	77
5.3	Preliminaries and Problem Definition . . . . .	78
5.3.1	Privacy Leakage of RL . . . . .	78
5.3.2	Problem Definition . . . . .	79
5.4	Differentially Private Deep Reinforcement Learning (DP-DRL) . . . . .	80
5.4.1	Overview of Methodology . . . . .	80
5.4.2	Definition of Score Function . . . . .	81
5.4.3	Dynamic Adjustment of Privacy Budget . . . . .	83
5.4.4	Algorithm of DP-DRL . . . . .	85
5.4.5	Discussion . . . . .	88
5.5	THEORETICAL PROOF ANALYSIS . . . . .	90
5.5.1	Differential Privacy Analysis . . . . .	90
5.5.2	Model Utility . . . . .	91
5.5.3	Time Complexity Analysis . . . . .	92
5.6	Experiments . . . . .	93

## TABLE OF CONTENTS

---

5.6.1	Experimental Environment . . . . .	93
5.6.2	Experimental Settings . . . . .	94
5.6.3	Performance Metrics . . . . .	96
5.6.4	Experimental Results . . . . .	97
5.7	Summary and Future Work . . . . .	103
<b>6</b>	<b>One Parameter Defense - Defending against Data Inference Attacks via Differential Privacy</b>	<b>105</b>
6.1	Introduction . . . . .	105
6.2	Related work . . . . .	108
6.2.1	Regularization-based defense methods . . . . .	109
6.2.2	Adversarial example-based defense methods . . . . .	109
6.2.3	Deep neural network-based defense methods . . . . .	109
6.2.4	Differential privacy-based defense methods . . . . .	110
6.2.5	Discussion of related work . . . . .	111
6.3	Preliminaries . . . . .	112
6.3.1	Data inference attacks . . . . .	112
6.3.2	Phase 1: Modify the confidence score vector . . . . .	114
6.3.3	Phase 2: Normalizing the confidence score vector . . . . .	115
6.3.4	Discussion of the method . . . . .	116
6.4	Properties of the defense method and how to use them to defend against attacks . . . . .	117
6.4.1	Privacy analysis . . . . .	117
6.4.2	Analysis of tuning $\epsilon$ . . . . .	120
6.4.3	Defending against membership inference attacks . . . . .	125
6.4.4	Defending against model inversion attacks . . . . .	126
6.5	Experiments . . . . .	126
6.5.1	Experimental setup . . . . .	126
6.5.2	Experimental results . . . . .	129
6.6	Summary and Future Work . . . . .	134
<b>7</b>	<b>Conclusion</b>	<b>137</b>
	<b>Bibliography</b>	<b>139</b>

## LIST OF FIGURES

FIGURE	Page
3.1 A real-world example: all students and teaching staff of a course need to execute their assignments on a cloud computing servers, but they have different choices available to them. . . . .	19
3.2 Overview of Scenario setting . . . . .	20
3.3 Overview of Differentially Private Advising Framework . . . . .	24
3.4 Experiment 1: Comparison between three strategies . . . . .	33
3.5 Experiment 1: Performance Comparison Between three strategies (Traditional Reinforcement Learning as base) . . . . .	33
3.6 Experiment 2: Comparison between three strategies with the same setting .	34
3.7 Experiment 2: Performance Comparison Between three strategies (Traditional Reinforcement Learning as base) . . . . .	34
3.8 Experiment 3: Comparison between different epsilon value with the same setting . . . . .	34
3.9 Experiment 3: Performance Comparison Between different epsilon value (Traditional Reinforcement Learning as base) . . . . .	34
4.1 An example of a logistic map . . . . .	43
4.2 Unit $a$ 's local area . . . . .	43
4.3 A high-level logistic map from agent $a$ 's perspective . . . . .	49
4.4 Obfuscation of agent $b$ 's local map . . . . .	50
4.5 A logistic map created by obfuscated local maps . . . . .	55
4.6 A high-level map featuring relative distances from agent $a$ 's perspective . . .	56
4.7 Performance of the four approaches on the logistics scenario with variation of the map size . . . . .	66
4.8 Performance of the <i>DP-based</i> approach on the logistics scenario with variation of the privacy budget value . . . . .	68

LIST OF FIGURES

---

4.9	Performance of the four approaches on the packet routing scenario with variation of the network size . . . . .	68
4.10	Performance of the <i>DP-based</i> approach on the packet routing scenario with variation of the privacy budget . . . . .	69
4.11	Performance of the three approaches on the packet routing scenario with variation of the dynamism . . . . .	69
5.1	An example of trajectory disclosure from a well-trained DRL agent . . . . .	74
5.2	Overview of DP-DRL . . . . .	81
5.3	Process of the score function . . . . .	82
5.4	Example maps in different sizes . . . . .	95
5.5	NRR vs populations at varying privacy budgets . . . . .	100
6.1	An input data record is accurately reconstructed by an attack model which has never seen this data record. . . . .	107
6.2	Overview of our method. The method consists of four steps. In Step 1, the range $[0, 1)$ is divided into $k$ non-overlapping sub-ranges, where $k$ is the number of classes that the target model $T$ can classify. In Step 2, each sub-range is matched with a score in vector $\mathbf{y}$ , where $\mathbf{y}$ is output by $T$ , ensuring that a sub-range with smaller values is matched with a smaller score. In Step 3, each sub-range is uniformly discretized into a set with $m$ scores, where $m$ is a hyper-parameter. Then, for each sub-range $i$ , the exponential mechanism is used to select a score from $m$ scores. The selected score is named $y'_i$ used to replace $y_i$ . In Step 4, since it is likely that $\sum_{i=1}^k y'_i \neq 1$ , the exponential mechanism is used again to normalize vector $\mathbf{y}'$ to $\mathbf{z}$ which is shown to the attacker. . . . .	113
6.3	Defenses against the model inversion attacks on MNIST. Rows 3-6 show the results for our DP-based method with different $\epsilon$ values. . . . .	131
6.4	Defenses against the model inversion attacks on Fashion-MNIST. Rows 3-6 show the results for our DP-based method with different $\epsilon$ values. . . . .	132
6.5	Defenses against the model inversion attacks on CIFAR10. Row 3 shows the results for our DP-based method with $\epsilon = 0.1$ . . . . .	133
6.6	Classification accuracy with varying $\epsilon$ values . . . . .	133
6.7	Confidence score distortion with varying $\epsilon$ values . . . . .	134
6.8	Membership inference accuracy with varying $\epsilon$ values . . . . .	134
6.9	Inversion error with varying $\epsilon$ values . . . . .	135

## LIST OF TABLES

<b>TABLE</b>	<b>Page</b>
2.1 Notations . . . . .	10
3.1 Agent $i$ 's Dataset of $Q$ -value . . . . .	28
3.2 Agent $j$ 's Dataset of $Q$ -value . . . . .	28
3.3 Agent $i$ 's Convergent $Q$ -value . . . . .	28
3.4 Agent $j$ 's Convergent $Q$ -value . . . . .	28
3.5 Parameter setting . . . . .	33
5.1 Defense Performance With Varying Privacy Budgets At Different Map Sizes .	98
5.2 Example of Recovered Grid Maps . . . . .	99
5.3 Model Utility Of Map Size $12 \times 11$ At Different Privacy Budgets . . . . .	101
6.1 Data allocation . . . . .	127
6.2 Comprehensive results of the three defense methods . . . . .	127





## INTRODUCTION

## 1.1 Background

**R**einforcement learning (RL) is a principled mathematical framework for autonomously experience-driven learning [112]. The primary goal of RL is to train autonomous agents to learn the optimal behaviors for their interactive environments. At each step, an RL agent observes the current state of the environment and takes an action, which causes a transition from the current state into a new state. The agent receives a scalar reward that evaluates the quality of the action it takes. The agent aims to find the optimal policy composed by a sequence of actions in the environment, which maximizes the total obtained reward.

Deep Reinforcement Learning (DRL) [78] promotes a higher-level understanding of the visual world in the field of RL by combining deep learning models and RL algorithms. Deep Q-network (DQN) is a typical structure in DRL by combining deep neural network and Q-learning approach in RL, which convolutional neural networks can be used as components of RL agents allowing them to learn directly from raw and high-dimensional inputs. At present, DRL algorithms are widely applied in the fields of video games [78] [117], autonomous robotics [66], and autonomous driving [100] [99] [6].

Multi-agent reinforcement learning (MARL) is a sub-field of RL, which studies the behaviors of multiple RL agents in a shared environment. In a MARL system, each agent is still motivated by its own rewards, however, an agent's behavior may affect other agents' behaviors, or be affected by others' behaviors, because the agent's observed states

include other agents' states and behaviors. MARL allows multiple agents exploring different alignments of their interests in the shared environment: cooperation and competition. In a cooperative environment, all agents learn towards a common goal with the exact same rewards. An agent is allowed to share learning experience with others if it meets some state many times before and feels confident enough to take the correct action. The way of such experience sharing is called the advising framework which accelerates agents learning by reusing previous knowledge in current repeated state from other agents [134]. On the other hand, the agents play against each other, and are motivated by the opposite rewards to each other in a competitive environment. The competitive MARL system is widely applied to revolve around social dilemmas in game theory problems, such as prisoner's dilemma and chicken and stag hunt.

## 1.2 Privacy Concerns

Since RL is achieving great success in an increasing number of application fields that may involve huge amounts of private information, the security of policies and privacy preservation in RL have given rise to widespread concerns. DRL policies are parameterized by neural networks which have been demonstrated to be vulnerable to adversarial attacks in supervised learning settings [50]. For instance, some perturbations added to the input can cause the convolutional neural networks to misclassify the adversarial input unexpectedly [44, 113], and cause negative impacts on the utility of trained models. Meanwhile, machine learning algorithms have been found possible to leak private information of individual training data in the past few years, which may contain users' sensitive information [95]. Membership inference attack [106] is possible to determine if some individual data points were used to train the model in a black-box setting.

Some attack methods also challenge the security of policies in DRL. DRL policies can be attacked under different goals such as reducing the final rewards of agents, or malevolently luring training agents to dangerous states [67]. Most of existing approaches [42, 50, 60] based on adversarial examples that allow the adversaries to make only limited changes to the raw inputs. Unlike supervised learning involving a fixed dataset of training examples during learning, DRL gathers input examples containing observations and states information throughout the training process dynamically. Therefore, in domains where an adversary can directly modify the training agent's input, adversarial perturbations are injected on frequently during DRL training. Otherwise, adversarial policy can be applied to maliciously change the training agent's physical observations, in

order to inject adversarial information into the agent’s input.

Privacy leakage also occurs in MARL systems, which involve multiple agents collaboratively explore in a shared environment. In a MARL system, agents’ actions or behaviors are directly exposed to other agents. Agents can be observed by other agents as a part of states, which further influence other agents’ selections of actions. In some scenarios, agents are even required to share with others the learning experience and the information in the environment which they have met before. Some privacy concerns have been raised in MARL when multiple agents come from different parties, or agents’ learning experience and policies are private to each other. For example, research into planning problems in MARL largely focuses on jointly automated planning [118]. During jointly automated planning, agents have to share information, which often results in the leaking of agents’ private information.

The privacy concerns do not only exist inside of MARL systems, but also in some scenarios where MARL algorithms are applied. As MARL algorithms are powerful tools to learn policies in many tasks, they are also some attackers’ weapons to execute some cyber attacks. Cyber attacks are typical game theory scenarios where attackers and defenders are two-party agents of MARL algorithms to play against each other. Attackers aim to learn how to beat defenders’ policies and maximize their malicious interests, meanwhile, defenders want to maximally protect their sensitive data from being attacked by attackers. Such scenarios are undoubtedly challenging the security and privacy of MARL agents. Therefore, with the continuous developments of RL in increasing application fields, it is essential to address the privacy and security risks in RL systems and algorithms.

### 1.3 Research Objective

Our research objective is to address privacy concerns in the RL environment, especially the scenarios we discussed above. We contend that differential privacy (DP) [29] has the potential to provide a privacy-preserving solution in RL. DP introduces the concept of *neighboring datasets*, which are two datasets that differ by only one record. DP offers a strong privacy guarantee through the addition of perturbation to a pair of datasets, meaning that the results of queries of two neighboring datasets are likely to be statistically similar [103].

DP has several attractive properties that make it quite valuable for RL. First, DP provides a strong privacy guarantee by hiding individuals in aggregated information.

DP can add some uncertainty onto agents' private information during the training, and reduce the impact of malicious agents in most tasks. Secondly, DP protects a higher-level model that captures behaviors rather than just limiting itself to a particular data point, which benefits from its stability that the probability of any outcome from RL algorithm is statistically unchanged by modifying on any individual record in data. Thirdly, DP has a great flexibility in RL applications where targets that need to be protected can be varied. The perturbation from DP can be added on any part of agents' learning in RL systems as required. Therefore, we aim to apply DP to benefit RL in the privacy preservation and security of varied application scenarios.

## 1.4 Main Contributions

This section will introduce the following four contributions in this thesis.

### 1.4.1 Differentially Private Advising Framework in Simultaneously Reinforcement Learning

Advising frameworks address the above problem by reusing previous knowledge in a repeated state from other agents. Simultaneously learning agents advising can be used to accelerate learning when all agents start learning in a multiple-state system at the same time [23, 104, 134, 137]. Some of the learning agents who have visited a state more times than others, can be seen as more experienced in. A single agent can play both roles of teacher who can provide advice, and student who will ask for advice.

However, traditional advising frameworks are only appropriate for the situation which all agents have the same actions, but cannot be applied when agents' actions are not the completely same. In real-world, it's commonly that agents have different available actions even though they are aiming the same task. We find that differential privacy property can guarantee that the experience for each action can still be used as advice if the two actions differ in, at most, one record. We propose a differentially private advising approach to improve the existing advising frameworks when agents' actions are different. This approach can expand the applicable field of advising frameworks and increase the probability of the occurrence of advising.

### **1.4.2 Differentially Private Reinforcement Learning Based Multi-agent Planning**

Multi-agent planning is one of the fundamental research problems in MARL systems [26, 135], which aims to improve agents' working efficiency by making plans in advance. Research into collaborative multi-agent planning largely focuses on jointly automated planning [118], where agents have to share information. However, this kind of information sharing often results in the leaking of agents' private information. Accordingly, to protect agents' privacy, privacy preservation is introduced into the collaborative multi-agent planning process [101, 102]. The main problem associated with privacy preservation in collaborative multi-agent planning is that of how to make plans for agents while also preserving the privacy of each agent.

To address the privacy preservation in multi-agent planning process, we develop a novel strong differentially private planning approach for distributed and communication-constrained environments. Our approach focuses primarily on logistic-like problems, i.e. planning, routing and scheduling problems exploring solutions that how robotics agents can efficiently deliver items from one city to another, which are typically used as running examples in multi-agent planning problems. In our proposed approach, the privacy budget can naturally be used to control communication overhead, with the result that only a limited number of messages are permitted during a planning phase.

### **1.4.3 Privacy Preservation in Deep Reinforcement Learning: a Training Perspective**

It has been demonstrated that DRL can leak private information about the training environment. An illustrative example [86] concerns an agent that aims to navigate the shortest path between a starting point and a destination in a simple grid world with obstacles. A well-trained agent will still follow the same trajectory even once all obstacles in the environment have been removed after training. This example indicates that it is possible to infer environmental information from a well-trained DRL policy. A DRL agent tends to memorize the training environment instead of performing visual navigation.

We contend the root cause of privacy leakage in DRL is the agent's observations of the training environment. When the agent is brought into a new open environment, the agent repeats the sequence of actions from memory, which means that private information concerning its trajectory and the original environment can be inferred. To solve the above issues, we propose the differentially private DRL method to protect a DRL agent's

training environment information against privacy leakage attacks, which is the first work to defend such attacks. We apply the exponential mechanism of DP to protect an agent’s observations from each visited state by obfuscating one observation element each time based on a probability distribution. Our method can also dynamically adjust the privacy budget to guarantee the privacy for both the agent and the training environment, as well as the utility of the trained agent’s policy.

#### **1.4.4 One Parameter Defense - Defending against Data Inference Attacks via Differential Privacy**

Our proposed differentially private DRL framework acts on the neural networks of DRL algorithms, thus, the similar idea can also be applied in classification problems in deep learning. We develop a time-efficient defense method against both membership inference and model inversion attacks in deep learning. We are the first to propose a one-parameter defense method that requires only one parameter to be tuned, the privacy budget. Our solution is a differential privacy mechanism that modifies and normalizes the confidence score vectors to confuse the attacker’s classifier. We theoretically demonstrate how to tune the privacy budget to defend against both types of attacks, while controlling the utility loss of confidence score vectors. We empirically show that the presented method effectively mitigates both types of attacks with no loss of classification accuracy, zero training time, and very low test time.

### **1.5 Thesis Organization**

The remainder of the thesis is organized as follows.

**Chapter 2** introduces preliminaries, mainly including the notations applied in this thesis, some definitions and properties on RL and DP, respectively.

**Chapter 3** introduces our proposed differentially private advising framework in simultaneously RL systems. The chapter includes the algorithms of the proposed method and demonstrates the performance in its experiment settings.

**Chapter 4** proposes a differentially private RL based multi-agent planning framework to protect the agents’ privacy when multiple agents’ collaborative plan an optimal policy in a RL system. We provide our algorithms and proofs, and demonstrate the effectiveness of our method in logistic-like scenarios.

**Chapter 5** and **Chapter 6** demonstrate how the exponential mechanism of DP to defense against privacy inference attacks to DRL and deep learning, respectively. In **Chapter 5**, we introduce a differentially private DRL framework to defense against privacy leakage attacks in grid worlds as examples. We provide our theoretical proof, and demonstrate the performance of the defense method against privacy leakage attacks in multiple-sized grid maps via different settings of the privacy budget. In **Chapter 6**, a similar method applying the exponential mechanism of DP is introduced to defense against both model inversion attacks and membership inference attacks in classification problems in deep learning settings.

**Chapter 7** summarizes the contributions of this thesis.





## PRELIMINARIES OF REINFORCEMENT LEARNING AND DIFFERENTIAL PRIVACY

In this chapter we will introduce the background knowledge for reinforcement learning and differential privacy, respectively. Some notations used in this thesis will be introduced first in the beginning of this chapter. We will then introduce the basic theory of traditional reinforcement learning with emphasis on  $Q$ -learning, and deep reinforcement learning, which is the combination of reinforcement learning algorithms and artificial neural networks. After that, we will introduce the general theories of DP.

### 2.1 Notations

Table 2.1 summarizes some major notations used in this thesis. Some other symbols used temporarily in different chapters, will be introduced in corresponding chapters.

### 2.2 Reinforcement Learning

RL is a principled mathematical framework for autonomously experience-driven learning [112]. In RL an autonomous agent tries to solve a task in an interactive environment which is unknown to the agent. The agent can shift the state of the environment by taking actions, and receive immediate feedback for each action from the environment.

Categories	Notations	Explanation
RL related	$\mathcal{S}$	A set of states
	$s$	A state
	$\mathcal{A}$	A set of actions
	$a$	An action
	$\mathcal{T}$	Transition dynamics
	$\pi$	An agent's trained policy
	$\mathcal{O}$	A set of observations
	$o$	An observation element
	$\mathcal{R}$	A reward function
	$r$	A reward
	$\gamma$	A discount factor
	$Q$	A $Q$ function or a $Q$ value
	$t$	Time, time sequence or iterative round
	$\alpha$	Learning rate, or accuracy parameter
DP related	$\mathcal{M}$	A randomized algorithm
	$D$	A dataset
	$D'$	A neighboring dataset
	$f$	A query
	$\hat{f}$	A noisy output of a query
	$F$	A set of queries
	$\Delta S$	Sensitivity
	$\epsilon$	Privacy budget
	$\delta$	Accuracy parameter

Table 2.1: Notations

The primary goal of the agent is to learn the optimal behavior with an optimal chain of actions in this environment.

RL is one popular area within machine learning, but it is fundamentally different from traditional machine learning methods in several aspects. First, traditional machine learning methods strongly depend on data acquisition, however, an RL agent learns from its own experience for the interaction with the environment and does not rely on supervision. Second, an RL agent focuses on searching an optimal behavior rather than analyzing any data. Third, an RL agent can be affected by the sequence of its experience and actions it takes. Instead, the order of data cannot affect the training of traditional machine learning methods.

RL techniques are typically used to solve sequential decision-making problems, which model the learning environment as a Markov decision process (MDP). An MDP model is

depicted as a tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ , where

- $\mathcal{S}$  is a set of states;
- $\mathcal{A}$  is the set of actions available to the agent;
- $\mathcal{T}$  is the transition dynamics defined as a probability mapping from state-action pairs to states  $\mathcal{T} : (\mathcal{S} \times \mathcal{A}) \times \mathcal{S} \rightarrow [0, 1]$ ;
- $\mathcal{R}$  is a reward function representing the benefit of performing an action  $a$  in the state  $s$ ;
- $\gamma$  is a discount factor applied to discount future rewards for reward accumulation, which satisfies  $0 \leq \gamma \leq 1$ .

The aim of RL is to find an optimal policy  $\pi$  that maps state-action pairs to a probability distribution  $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ , in order to maximize the obtainable reward  $\mathbb{E}_\pi[\sum_{p=0}^P \gamma^p r_p]$ , where  $P$  is state-action pairs chosen by the agent from the beginning of a training epoch to the point of its success or failure.

### 2.2.1 Traditional Reinforcement Learning

Traditional RL is driven by  $Q$  learning, also known as *temporal difference method* (TD).  $Q$  learning is a value-function based algorithm [97] which uses TD evaluation of the  $Q$  value to update the estimated  $Q$  function. The update rule is shown as follows:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a')];$$

where  $s'$  is the next state, and  $a'$  is an action of the next state  $s'$ , which currently has the largest  $Q$  value. The traditional  $Q$  learning algorithm is shown in **Algorithm 1**.

There are some limitations in traditional RL algorithms because all the values are stored in  $Q$  tables. First, the tabular structure limits the number of states and actions to a very small scale, but in the real-world problems, the state space can be too large to be stored into a table due to the computer's storage and capability. Second, the tabular structure is hard to learn from the previous experience of a similar state and share knowledge. To improve the traditional RL algorithms and overcome above restrictions, function approximators are applied to replace tables to map states of the training environment. DRL is the most common method for function approximation. DRL extracts relevant features from inputs directly that generalize to unseen states.

---

**Algorithm 1**  $Q$  learning

---

**Require:** Initialize:  $Q$ -value for each actions, initial state  $s = s_0$

```

1: for each epoch do
2:   while  $s$  is NOT terminal do
3:      $a \leftarrow \pi(s)$ 
4:      $r \leftarrow \mathcal{R}(s, a)$ ;
5:      $a' \leftarrow \max_{\tilde{a}} (Q(s', \tilde{a}))$ 
6:      $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma Q(s', a')]$ ;
7:      $s \leftarrow s'$ ;
8:   end while
9: end for

```

---

## 2.2.2 Deep Reinforcement Learning

DRL [78] promotes a higher-level understanding of the visual world in the field of RL by combining deep learning models and RL algorithms. DeepMind presented DQN that combined  $Q$  learning with neural networks. DRL algorithms perform well on video games directly from pixels, such as Atari [78] and Mujoco [117], and autonomous robotics from camera inputs in the fields of robotic control tasks [66] and autonomous driving [99, 100] in the real world [6].

The DQN algorithm involves two novel mechanisms, *experience replay* and *frozen target network*. *Experience replay* can store agent's experience  $\langle s_t, a_t, r_t, s_{t+1} \rangle$  as a tuple in a buffer. At each training step, the agent uniformly sample a mini-batch of experience tuples from the buffer to update the network. Such mechanism significantly reduce the correlation between experience samples without overfitting in the network. Also, reusing previous experience is also beneficial to an DQN agent for a smooth and efficient learning. Meanwhile, *Frozen target network* uses two networks with an identical architecture but different weights values for  $Q$  network and the target network, which are known as  $\theta$  and  $\theta^-$ , respectively. The target network stays frozen and only updates itself at every  $C$  steps by copying the parameters of  $Q$  network, where  $C$  is a predefined constant. This mechanism leads to a more stabilized learning by reducing the oscillation of a DQN agent's policy. The **Algorithm 2** describes the process of the DQN algorithm with experience replay and frozen target model mechanisms.

**Algorithm 2** DQN algorithm with experience replay and frozen target model

---

**Require:** Initialize replay memory  $D$  with capacity  $N$ , action-value function  $Q$  with random weights  $\theta$ , target action-value function  $\hat{Q}$  with weights  $\theta^- = \theta$ , initial state  $s_0$ , training maximum epochs  $E$ , current training epoch  $e$ , the constant  $C, \epsilon_g$ ;

- 1: **for**  $e = 1$  to  $E$  **do**
- 2:      $s = s_0$ ;
- 3:     **while**  $s$  is NOT terminal **do**
- 4:         Get observations  $\mathcal{O}(s)$  at  $s$ ;
- 5:         Randomly get probability  $p, p \in [0, 1]$ ;
- 6:         **if**  $p > \epsilon_g$ , where  $\epsilon_g$  is threshold of greedy policy **then**
- 7:             Randomly select action  $a, a \in \mathcal{A}$ ;
- 8:         **else**
- 9:              $a \leftarrow \operatorname{argmax}_{\mathcal{A}} Q(\mathcal{O}'(s), \mathcal{A}; \theta)$ ;
- 10:         **end if**
- 11:         Obtained reward  $r \leftarrow \mathcal{R}(s, a)$ ;
- 12:         The next state  $s' \leftarrow \mathcal{T}(s, a)$ ;
- 13:         The transition  $t \leftarrow (\mathcal{O}'(s), a, r, \mathcal{O}'(s'))$ ;
- 14:          $D \leftarrow D \cup t$ ;
- 15:          $s = s'$ ;
- 16:         **if**  $e$  is divided by  $C$  **then**
- 17:             Sample random mini-batch of transitions  $(\mathcal{O}'_j, a_j, r_j, \mathcal{O}'_{j+1})$  from  $D$ ;
- 18:             Set  $y_j \leftarrow \begin{cases} r_j, & \text{if next step } j+1 \text{ terminates the epoch} \\ r_j + \gamma \max_{\mathcal{A}} \hat{Q}(\mathcal{O}'_{j+1}, \mathcal{A}; \theta^-), & \text{otherwise} \end{cases}$ ;
- 19:             Gradient descent on  $(y_j - Q(\mathcal{O}'_j, \mathcal{A}; \theta))^2$  with  $\theta$ ;
- 20:             Reset  $\hat{Q} = Q$
- 21:         **end if**
- 22:     **end while**
- 23: **end for**

---

## 2.3 Differential Privacy

DP is a provable privacy concept conceived by Dwork et al. [29]. Its core premise, namely that the outputs of the queries on neighboring datasets should be statistically similar, is one of the strongest standards for a privacy guarantee. The formal definitions of neighboring dataset and DP are presented below.

**Definition 2.1** (Neighboring Dataset). The datasets  $D$  and  $D'$  are neighbouring datasets if and only if they differ in one record. This is denoted as  $D \oplus D' = 1$ , where ' $\oplus$ ' indicates the difference between two datasets.

**Definition 2.2** ( $(\epsilon, \delta)$ -DP). A randomized algorithm  $\mathcal{M}$  gives  $\epsilon$ -DP for *neighboring datasets*  $D$  and  $D'$ , and for every set of outcomes  $\Omega$ ,  $\mathcal{M}$  satisfies:

$$(2.1) \quad \Pr[\mathcal{M}(D) \in \Omega] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(D') \in \Omega] + \delta .$$

For a particular output, the ratio on two probabilities is bounded by  $e^\epsilon$ .  $\epsilon$  is the privacy parameter, also known as the *privacy budget* [27], which controls the level of privacy preservation of a mechanism  $\mathcal{M}$ . If  $\delta = 0$ , the randomized mechanism  $\mathcal{M}$  provides  $\epsilon$ -DP by its strictest definition, which is usually called *pure DP*.  $(\epsilon, \delta)$ -DP gives freedom to violate strict  $\epsilon$ -DP for some low probability events, which is called *approximate DP* [8]. A smaller  $\epsilon$  means greater privacy.

The maximum difference on the queried results over neighboring datasets is defined as the sensitivity  $\Delta S$ . The formal definition is as follows:

**Definition 2.3** (Sensitivity). For a query  $f : D \rightarrow \mathbb{R}$ , the global sensitivity of  $f$  is defined as follows:

$$(2.2) \quad \Delta S = \max_{D, D'} \|f(D) - f(D')\|_1 .$$

**Definition 2.4** (Private prediction interface [28]). A prediction interface  $\mathcal{M}$  is  $\epsilon$ -differentially private, if for every interactive query generating algorithm  $f$ , the output  $(f \rightleftharpoons \mathcal{M}(F))$  is  $\epsilon$ -differentially private with respect to model  $F$ , where  $(f \rightleftharpoons \mathcal{M}(F))$  denotes the sequence of queries and responses generated in the interaction of  $f$  and  $\mathcal{M}$  on model  $F$ .

This definition shows that a private prediction interface can guarantee the privacy preservation of the interaction between queries and responses. Assume an attacker interacts with a prediction interface. If this interaction is differentially private, for each query of the attacker to the target model, he receives only an obfuscated response which breaks the relationship between the attacker's query and the corresponding response.

In DP, *Gaussian* and *Laplace* mechanisms are widely used for the numeric outputs of queries. These mechanisms add noise to the data to obscure certain sensitive attributes until others cannot distinguish the exact true answers of queries. The formal definitions are as follows:

**Definition 2.5** (Gaussian mechanism). The Gaussian mechanism with parameter  $\sigma$  adds zero-mean Gaussian noise with variance  $\sigma$ . Given a function  $f : D \rightarrow \mathbb{R}$  over a dataset  $D$ , if  $\sigma = S \sqrt{2 \ln(2/\delta)}/\epsilon$  and  $\mathcal{N}(0, \sigma^2)$  are independent and identically distributed (i.i.d.) Gaussian random variable, Equation 2.3 provides  $(\epsilon, \delta)$ -DP.

$$(2.3) \quad \hat{f}(D) = f(D) + \mathcal{N}(0, \sigma^2) .$$

**Definition 2.6** (Laplace mechanism). Given a function  $f : D \rightarrow \mathbb{R}$  over a dataset  $D$ , Equation 2.4 provides  $\epsilon$ -DP.

$$(2.4) \quad \hat{f}(D) = f(D) + \text{Laplace}\left(\frac{\Delta S}{\epsilon}\right).$$

The *exponential* mechanism allows a precise data record from a dataset while preserving DP by specifying a score function, which outputs a score for each element in the dataset. The formal definitions are as follows:

**Definition 2.7** (Score function). The quality of an outcome is measured by a score function  $q : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \rightarrow \mathbb{R}$ , where  $q(D, \phi)$  is a measure of how good an outcome  $\phi$  would be on dataset  $D$ .

**Definition 2.8** (Exponential mechanism). Let  $q(D, \phi)$  be a score function of dataset  $D$  that measures the quality of output  $\phi \in \Phi$ . Thus, an exponential mechanism  $\mathcal{M}$  satisfies  $\epsilon$ -DP when

$$(2.5) \quad \mathcal{M}(D) = \phi, \text{ with probability } \propto \exp\left(\frac{\epsilon q(D, \phi)}{2\Delta S}\right).$$

where  $\Delta S$  represents the *sensitivity* of the score function  $q$ .





## DIFFERENTIALLY PRIVATE ADVISING FRAMEWORK IN SIMULTANEOUSLY REINFORCEMENT LEARNING

Due to the rapid development of the cloud computing environment, it is widely accepted that cloud servers are important for users to improve work efficiency. Users need to know servers' capabilities and make optimal decisions on selecting the best available servers for users' tasks. We consider the process of learning servers' capabilities by users as a multi-agent reinforcement learning (MARL) process. The learning speed and efficiency in RL can be improved by sharing the learning experience among learning agents which is defined as advising. However, existing advising frameworks are limited by the requirement that, during advising all learning agents in an RL environment must have exactly the same actions. To address the above limitation, this chapter proposes a novel differentially private advising framework for MARL. Our proposed approach can significantly improve the application of conventional advising frameworks when agents have one different action. The approach can also widen the applicable field of advising and speed up RL by triggering more potential advising processes among agents with different actions.

### 3.1 Introduction

Regular RL approaches need a large number of interactions with the system environment to learn a policy [68]. Advising frameworks address the above problem by reusing

previous knowledge in a repeated state from other agents. Simultaneously learning agents advising can be used to accelerate learning when all agents start learning in a multiple-state system at the same time [23, 104, 134, 137]. An agent’s available choices under this state are called actions. Some of learning agents who have visited a state more times than others, can be seen as more experienced in. A single agent can play both roles of teacher who can provide advice, and student who will ask for advice.

**Figure 3.1** shows a real-world example, one assignment in a cloud computing course in a university. Both students and teaching staff need to execute this assignment on the university’s cloud servers, but students can only access public servers, and staff can access either public servers or private servers. In this example, the experienced students who have already executed their assignments more times than others can provide advice to other students in choosing the optimal server with the best capability to run their assignments within traditional advising frameworks, and the same applies to staff.

However, traditional advising frameworks are only appropriate for situations in which all agents have the same actions, and cannot be applied when agents’ actions are not exactly the same. In the real world, it is common that agents have different available actions even though they are solving the same problem [104]. Referring back to **Figure 3.1**, considering traditional frameworks, experienced teaching staff cannot advise students which server has a better capability and a better performance due to different available choices for servers even though they are working on the same assignment. Therefore, it is crucial to solve the problem of advising among agents with different available actions. Little previous work has considered experience transfer between two agents with different available actions, and the challenge is how to transfer the experience for the actions which are different among teachers and students.

We find that a DP mechanism can address the above challenge through the property of randomization. We consider all agents’ available actions and their experience on each state as independent datasets. The two datasets can be considered as neighboring datasets in terms of DP. In **Figure 3.1**, if teaching staff and students record the execution time of tasks as their experience on selecting servers, these records for servers can be seen as datasets. The less time means the better capability of servers. When there is a server accessible only to teaching staff, teachers’ datasets have one more record than students’. In this case, the datasets of teachers and students can be considered as neighboring datasets. By adding randomization, these datasets can be considered as the same in a defined scale, and teaching staff and students can share experience with each other. The DP property in our work is to guarantee that the experience for each action

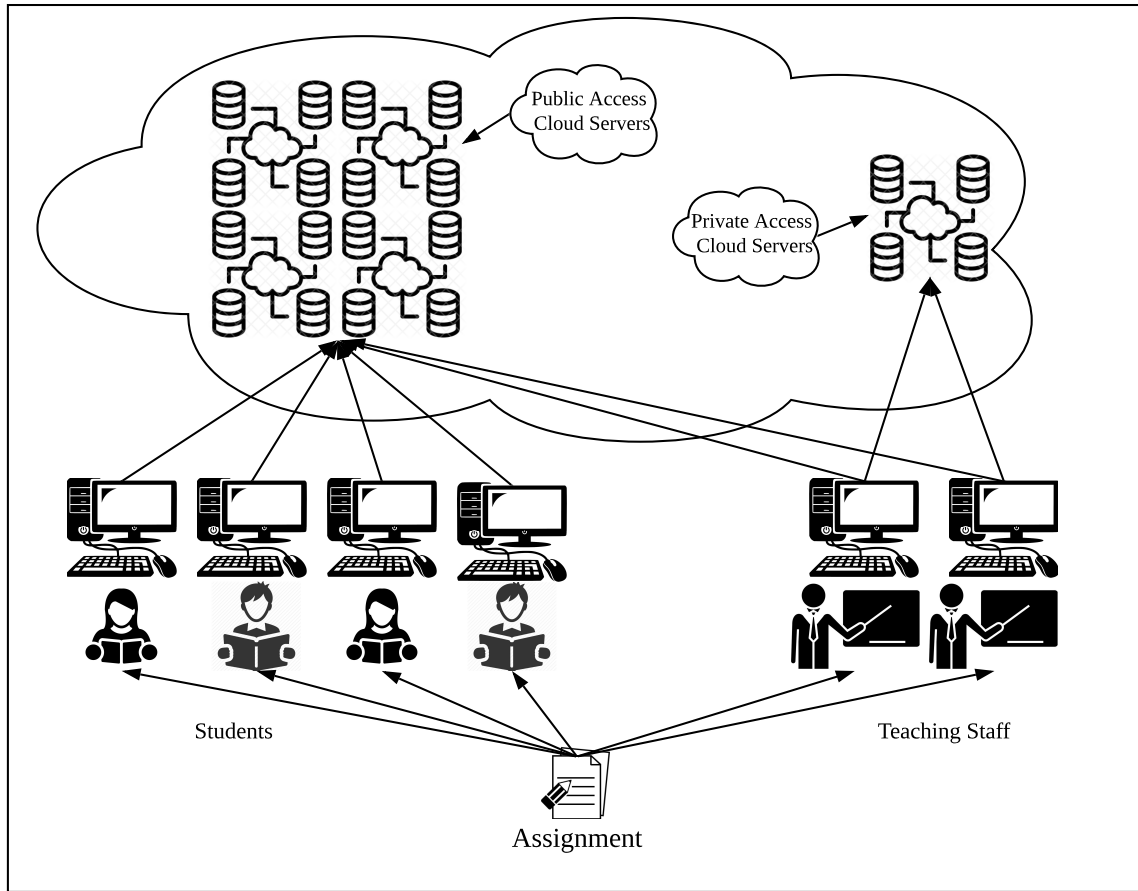


Figure 3.1: A real-world example: all students and teaching staff of a course need to execute their assignments on a cloud computing servers, but they have different choices available to them.

can still be used as advice if the two actions differ in, at most, one record. This approach can expand the applicable field of advising frameworks and increase the probability of the occurrence of advising. To summarize the three main contributions in this chapter:

- Firstly, we propose an improved differentially private advising approach based on our previous framework, to improve conventional advising frameworks to address the problem that agents' actions are different.
- Secondly, we widen the applicable field of conventional advising frameworks in RL by allowing more possible advising to accelerate agents' learning stage in RL.
- Thirdly, we propose a new approach to decide when advising happens by calculating the Euclidean distance of teacher and student's  $Q$ -value,  $Q(s)$ .

We design a comprehensive set of experiments to demonstrate our approach in the simultaneous MARL framework by comparing with conventional advising frameworks under the same setting. We provide a brief analysis of the convergence of agents' learning and why Euclidean distance works here. We also present our experiment results and the analysis of the performance of DP in the method by changing the value of privacy budget,  $\epsilon$ , in the range from 0.1 to 1.

## 3.2 Problem Statement

### 3.2.1 Scenario setting

In this section, we describe a detailed scenario as a motivated case study to demonstrate existing challenges we are addressing.

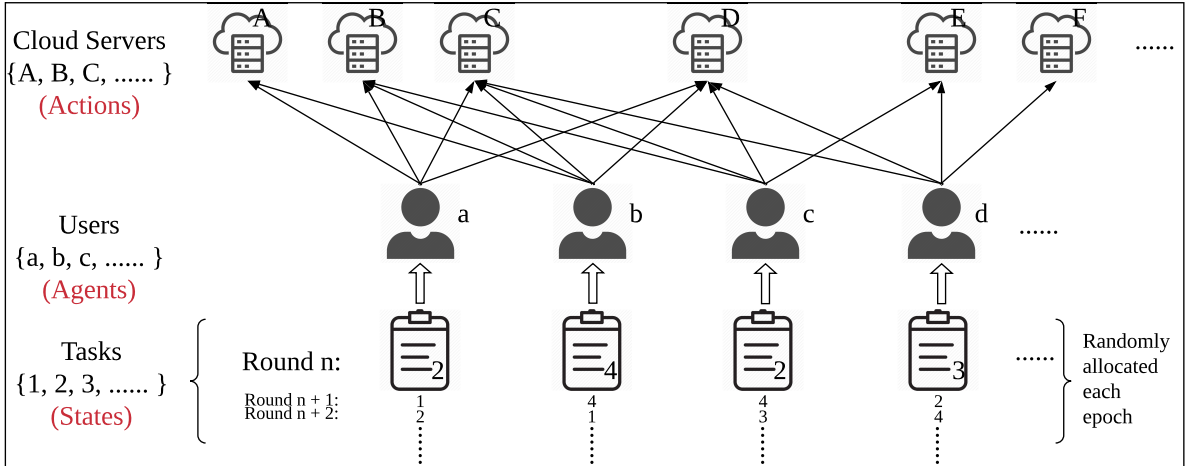


Figure 3.2: Overview of Scenario setting

The scenario shown in **Figure 3.2** is an extension of above example in a cloud environment. There are many users,  $a, b, c, d, \dots$ , whom we consider as **agents** participating in an RL environment. There are many tasks,  $1, 2, 3, 4, \dots$  can treat as the **states**  $s_1, s_2, s_3, s_4, \dots$  in the set of states  $\mathcal{S}$ . Each user is allocated to a random task each time, we call a single cycle for a user to finish its task an *epoch*, from being allocated to a random task to the time before being allocated to the next task. Cloud servers  $A, B, C, D, \dots$  are used to solve tasks for users. In this RL environment, servers are actions taken by agents to solve tasks. In our scenario, each agent can access different servers and compose a set of available actions,  $\mathcal{A}$ , i.e. for agent  $a$ ,  $\mathcal{A}_a(\mathcal{S}) : A, B, C, D$ . Each agent can

only take one action for each task each epoch. However, each server behaves differently on solving different tasks, and servers behave differently on the same task, i.e. server 'A' is the best server on executing task 2 with the lowest time cost but the worst on task 4, or computation capability of the server 'C' is not enough to solve task 3. Combining all these factors, a user obtains feedback to evaluate how this server executes this task, which we call reward  $r(s, a)$ , meaning the reward from the action  $a$  at the state  $s$ . A higher reward means a server has a better capability on solving the task, and vice versa. Therefore, the aim of users is to solve the task with the optimal server with the maximum reward each epoch.

### 3.2.2 Scenario discussion

In the above scenario, each user should repeatedly be allocated tasks and choose servers to solve them. The process can be considered as a learning process in which users need to learn capabilities of servers on solving tasks. We explore the problems of how users can learn servers' capabilities, whether users can share their experience to others during the learning process, and how users share the experience to others even they have different available choices.

To address these questions, each agent needs to know each cloud server's capabilities based on their previous sequence of learning experience, in order to make the best decision next time being allocated to the same tasks. Users continuously expand their knowledge by receiving the rewards from previous epochs, and use learned knowledge to make the next decision. The knowledge determines probability distribution for each agents' available actions in each state. The agent should 'understand' this reward and update their probability distribution for available actions at states by a transition function  $\mathcal{T}$ , in order to make a better decision next time when they are allocated to this task.

Agents can transfer experience to or ask experience from others to accelerate the learning process. However, due to the randomness of the task allocation to simultaneously learning agents, the above scenario is very likely to generate a knowledge gap at some state. The knowledge gap is an experience gap at a state, based on difference of times that an agent is allocated to a task with others. An agent can be considered as more experienced than another on a task if this agent has met this task enough times more than the other. A threshold is used to define 'more experienced' with a minimum number of knowledge gap. An agent can ask for advice from the experienced agent while being allocated to this task again. However, agents will inevitably run into due to the limitation

we mentioned above, i.e. traditional advising frameworks only focus on knowledge transfer among agents with the same actions. For instance, referring back to Figure 3.2, agent  $a$ 's available actions are  $\mathcal{A}_a : \{A, B, C, D\}$ , and agent  $c$ 's are  $\mathcal{A}_c : \{B, C, D, E\}$ . In conventionally proposed advising frameworks, agents can only advise other agents with the same actions. In other words, agent  $a$  and  $c$  cannot advise each other due to the difference in their actions sets. Therefore, we propose a differentially private advising framework to break this limitation. Our approach allows agents to advise in the above scenario when RL agents have different available actions, which will accelerate their learning.

### 3.3 Related Work

The early advising approaches mostly focused on humans as teachers to provide advice. Clouse and Utgoff[21] proposed that humans can offer advice at any time as an expert monitoring a single student learning a multiple-step decision task via an RL. Maclin and Shavlik[69] proposed that the teacher occasionally gives the suggestions to the student while watching the student's learning process. In 2005, Torrey et al. [122] proposed a method which requires a human-provided and hand-coded mapping to link the two tasks, to transfer knowledge from one task to another as advice. Clouse [20] proposed to train an RL agent using autonomous teacher that is assumed to perform at a moderate level of expertise for the task. However, this approach may restrict the performance of advising due to receiving too much advice. *Teacher-student* framework is firstly proposed by Torrey and Taylor [121] and further improved by Taylor et al. [116], which introduces a numeric communication budget to limit the number of times of advising during the learning steps, which becomes an essential part of the advising model to imitate humans' availability and attention capability in real-world [77]. Zimmer et al. [146] introduced an approach that an agent should learn when to give advice with building a sequential decision-making problem. Even though internal representations of the student are not supposed to be known, the teacher must observe the student reward to solve the problem. All the aforementioned works rely on a single teacher with a fixed policy. Nunes and Oliveira [84] proposed that multiple agents can broadcast their average reward at the end of each learning epoch while learning in the same system, and agents whose average reward is lower than the best one can ask for device from the best agent. Zhan et al. [139] considered the possibility of receiving sub-optimal advice, and combined multiple bits of advice by a majority vote to make this method more robust to against bad advising.

In above approaches, either teacher or student alone can trigger a process of advising in RL stage. Amir et al. [3] proposed a jointly-initiated framework that both students and teachers need to agree to receive and provide advice simultaneously. Da Silva et al. [23] proposed an advising framework among simultaneously learning agents based on *teacher-student* framework, in which agents start learning together without an experienced agent. Their method is also based on jointly-initiated teacher-student relations which are established on demand when a student is not confident enough to make choice alone, and a teacher has enough confidence to provide advice at the same time, which is highly related to our proposed method. Ye et al. [137] proposed to use DP to protect benign agents against malicious agents in the advising stage. Zhu et al. [142] proposed a partaker-sharer advising framework (PSAF) for cooperative agents under limited communication. The framework allows Q-value transfer from an agent who has visited a state more times to an agent who has visited a state fewer times.

### 3.4 Preliminary

Supposing that an agent has learned an optimal policy  $\pi$  for a specific task and become experienced, it can teach another agent which is beginning to learn the same task using this fixed policy. As the student learns, the teacher will observe each state  $s$  the student encounters and each action  $a$  taken by the student. In  $n$  of these states, the teacher can advise the student to take the 'correct' action  $\pi(s)$  with its own learning experience or policy. The teacher-student framework aims to accelerate a student's training process. However, a budget  $b$  can limit the advising process. The teacher cannot provide further advice when the budget  $b$  is spent. Therefore, it is critical to define when to give advice to accelerate student's learning.

### 3.5 Differentially Private Advising Framework

We have proposed a differentially private advising framework by combining RL, teacher-student advising framework, DP mechanism and Euclidean distance together. The **Figure 3.3** is the overview of our proposed method. An RL agent  $j$  wants to update its probability distribution by obtaining a reward from an action. Any reason leading to the failure of advising results in the fact that agent  $j$  has to choose the action alone based on its own probability distribution. The agent  $j$  has a communication budget  $b_{ask}$  to control its communication overhead and each successful advising process costs

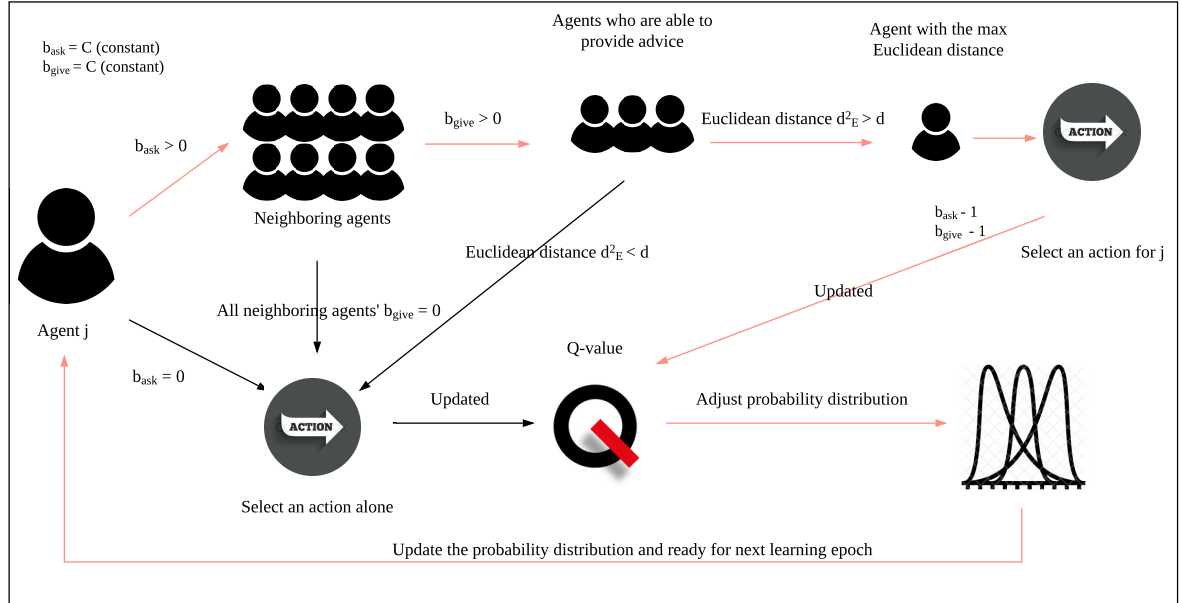


Figure 3.3: Overview of Differentially Private Advising Framework

some communication budget from it.  $b_{ask}$  is initialized with a constant  $C$ , and each ask consumes 1 from the budget. When it runs out the budget  $b_{ask}$ , it leads to the failure of advising. Agents can trigger advising with their neighbouring agents whose actions differ in at most one record with the agent. When  $b_{ask} > 0$ , agent  $j$  check if its neighbouring agents have communication budget  $b_{give}$  for giving advice.  $b_{give}$  is also initialized with a constant  $C$ , and each advice consumes 1 from the budget. If all neighbouring agents run out their  $b_{give}$ , the advising process fails. If some of neighbouring agents'  $b_{give} > 0$ ,  $j$  starts to calculate the Euclidean distance with each available neighbouring agent's  $Q$ -value. None of Euclidean distance results are greater than a threshold  $d$  will lead to the failure of advising process. If one or more than one Euclidean distance is greater than  $d$ , the agent with the maximum distance with  $j$  will become  $j$ 's teacher to choose an action for  $j$  for further updating  $Q$ -value and probability distribution in this epoch. We will introduce details and algorithms for each part in following subsections.

### 3.5.1 Reinforcement Learning And Normalization

**Algorithm 3** introduces the RL algorithm with normalization. For each epoch, an agent  $j$  selects an action  $a_k \in A(j)$  based on its probability distribution over the available actions at state  $s$ . The agent  $g$  obtains a reward  $r$  by taking this action, and uses this reward to update its  $Q$ -value of the action  $a_k$  at state  $s$ , which this update is based on: 1)



**Algorithm 3** Reinforcement Learning with Normalization**Require:** /\* Taking agent  $j$  at state  $s$  as an example \*/**Require:** Initialize probability distribution,  $Q$ -value for each actions

---

```

1: repeat for each epoch
2:   Agent  $j$  chooses an action  $a_k$ , based on the probability distribution:  $\pi(s) = \langle \pi(s, a_1), \dots, \pi(s, a_n) \rangle$ ;
3:    $r \leftarrow \mathcal{R}(s, a_k)$ ;
4:    $Q(s, a_k) \leftarrow (1 - \alpha)Q(s, a_k) + \alpha [r + \gamma \max_a Q(s', a)]$ ;
5:    $\bar{r} \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(s, a) Q(s, a)$ ;
6:   for  $a \in \mathcal{A}(s)$  do
7:      $\pi(s, a) \leftarrow \pi(s, a) + \xi (Q(s, a) - \bar{r})$ ;
8:   end for
9:    $\pi(s) \leftarrow \mathbf{Algorithm\ 2}(\pi(s))$ ;
10:   $s \leftarrow s'$ ;
11: until finish learning

```

---

$Q(s, a_k)$ , the current value of  $Q(s, a_k)$ , 2)  $\max Q(s', a)$ , the maximum  $Q$ -value of an action at the next state  $s'$ , 3)  $\gamma$ , a discount rate to control the effect of  $\max Q(s', a)$  on update this time, 4)  $\alpha$ , a learning rate to control the speed of updating  $Q$ -value and learning, and 5) the reward  $r$ . An agent should to adjust its probability distribution using the updated  $Q$ -value. Similar to the process of updating  $Q$ -value, the agent updates its probability distribution  $\pi(s)$  for each action based on the current  $\pi(s, a)$ , a learning rate  $\xi$ , and  $\bar{r}$  which is the sum of the product of each action's probability and  $Q$ -value. The update of  $\pi(s)$  can only show the trend of experience accumulation and reflect the relationship between each action's  $Q$ -value, therefore, we require normalization function shown in **Algorithm 4** to adjust the probability distribution  $\pi(s)$  to satisfy the requirement that the sum of  $\pi(s)$  is 1 while keeping the ratio relationship.

### 3.5.2 When To Ask For And Give Advice

In this section, we introduce when to ask for and give advice and why the DP mechanism is appropriate here. In our advising environment, agents' actions are not private information, which means every agent knows their neighbours and neighbours' actions. Therefore, neighbouring relationship has been defined for each client to trigger advising processes. We firstly present the convergence proof for  $Q$ -value in our case as a base, and explain why the vector  $Q(s, a)$  and our queries on the vector satisfy DP. The solution we used to decide the timing of advising is calculating Euclidean distance  $d_E^2(\text{teacher}, \text{student})$  between the student's  $Q$ -value vector and its optimal teacher's

---

**Algorithm 4** Normalization

---

**Require:** A vector  $p$  with the length  $n$

**Require:**  $d = \min_{1 \leq k \leq n} p(k)$ , mapping centre  $c_o = 0.5$  and mapping lower bound  $\Delta = 0.001$ ;

```

1: if  $d < \Delta$  then
2:    $\rho \leftarrow \frac{c_o - \Delta}{c_o - d}$ ;
3:   for  $k = 1$  to  $n$  do
4:      $p(k) \leftarrow c_o - \rho(c_o - p(k))$ ;
5:   end for
6: end if
7:  $\Pi \leftarrow \sum_{1 \leq k \leq n} p(k)$ ;
8: for  $k = 1$  to  $n$  do
9:    $p(k) \leftarrow \frac{p(k)}{\Pi}$ ;
10: end for
11: return  $p$ 

```

---

$Q$ -value vector. A threshold  $d$  is set that when the distance  $d_E^2(\text{teacher}, \text{student}) > d$  we consider the teacher to be sufficiently more experienced than the student to be able to advise them.

### 3.5.2.1 The Convergence Of $Q$ -value

Much of the previous research on the proof of  $Q$ -value has been exploratory in nature. We refer readers who are interested in the proof to [76, 125, 130]. In our case, due to the discreteness of our states and actions, it is not necessary to think about the optimal  $Q(s', a')$  from the next state  $s'$  while updating current state's  $Q(s, a)$ , or simply let  $Q(s', a') = 0$ . The equation of updating  $Q$ -value is transformed to

$$\begin{aligned} Q(s, a) &\leftarrow Q(s, a) + \alpha \cdot (r(s, a) - Q(s, a)) \\ &\leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot r(s, a) \end{aligned}$$

**Lemma 3.1.** *Assume that an agent has selected the action  $a$  whose reward is  $r(s, a)$ , at the state  $s$  enough times, and updated its  $Q$ -value  $Q(s, a)$  until full convergence, we have*

$$\lim_{t \rightarrow +\infty} Q(s, a) = r(s, a)$$

**Proof.** Let  $q = Q(s, a)$ ,  $r = r(s, a)$ , and  $t$  is the index of times for updating  $q$ , so,  $q_t$  is the current value of  $Q(s, a)$ . When the  $(t + 1)^{th}$  time to update  $q$ , we have

$$q_{t+1} = (1 - \alpha) \cdot q_t + \alpha \cdot r$$

where  $\alpha$  is learning rate in the range  $0 < \alpha < 1$ . We have

$$\begin{aligned} q_{t+1} - r &= (1 - \alpha) \cdot q_t + \alpha \cdot r - r \\ &= (1 - \alpha) \cdot (q_t - r) \end{aligned}$$

So we have

$$\frac{q_{t+1} - r}{q_t - r} = 1 - \alpha$$

So  $(q_t - r)$  is a geometric progression with the common ratio  $(1 - \alpha)$ , we have

$$q_t = (q_0 - r) \cdot (1 - \alpha)^t + r$$

Because when  $0 < \alpha < 1$

$$\lim_{t \rightarrow +\infty} (1 - \alpha)^t = 0$$

So we have

$$\lim_{t \rightarrow +\infty} q_t = (q_0 - r) \cdot 0 + r = r$$

■

### 3.5.2.2 Differential Privacy

Assume that all agents have learned enough times, and  $Q(s, a)$  and its probability distribution are fully convergent, for the state  $s$  and the action  $a$ , the value of  $Q(s, a)$  is fixed and  $Q(s, a) = r(s, a)$  for each agent. Consider a single agent,  $i$ 's  $Q$ -value, who has  $n$  available actions,  $Q(s) : \{Q(s, a_1), Q(s, a_2), \dots, Q(s, a_n)\}$ . We use a table to record  $i$ 's  $Q(s)$  value as a dataset as shown in **TABLE 3.1**. Similarly, an agent  $j$ 's  $Q(s)$  value as shown in **TABLE 3.2**. When  $Q(s, a)$  values of agent  $i, j$  are fully convergent, **TABLE 3.1** and **TABLE 3.2** is then transformed to **TABLE 3.3** and **TABLE 3.4**, separately. The difference between **TABLE 3.3** and **TABLE 3.4** has been highlighted where agent  $i$  has a different action  $a_k$  at the state  $s$  from agent  $j$ 's action  $a_k^*$ , and accordingly different convergent  $Q(s, a_k)$  and  $Q(s, a_k^*)$ . According to **Definition 2.1**, we consider two datasets in **TABLE 3.3** and **TABLE 3.4** are neighboring datasets which satisfy DP mechanism. Because  $Q(s)$  maps the probability distribution  $\pi(s)$ , we consider  $\pi(s)$  is a query on the dataset  $Q(s)$ . We define the query  $f(Q(s))$  is the process of calculating  $\pi(s)$  from  $Q(s)$ . According to **Definition 2.2**, existing a privacy budget  $\epsilon$  and a randomized algorithm  $\mathcal{M}$  satisfy

$$Pr[\mathcal{M}(Q_i(s)) \in \Omega] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(Q_j(s)) \in \Omega]$$

Action	$Q(s)$
$a_0$	$Q(s, a_0)$
$a_1$	$Q(s, a_1)$
$\vdots$	$\vdots$
$a_k$	$Q(s, a_k)$
$\vdots$	$\vdots$
$a_n$	$Q(s, a_n)$

Table 3.1: Agent  $i$ 's Dataset of  $Q$ -value

Action	$Q(s)$
$a_0$	$Q(s, a_0)$
$a_1$	$Q(s, a_1)$
$\vdots$	$\vdots$
$a_k^*$	$Q(s, a_k^*)$
$\vdots$	$\vdots$
$a_n$	$Q(s, a_n)$

Table 3.2: Agent  $j$ 's Dataset of  $Q$ -value

Action	$Q(s)$
$a_0$	$r(s, a_0)$
$a_1$	$r(s, a_1)$
$\vdots$	$\vdots$
$a_k$	$r(s, a_k)$
$\vdots$	$\vdots$
$a_n$	$r(s, a_n)$

Table 3.3: Agent  $i$ 's Convergent  $Q$ -value

Action	$Q(s)$
$a_0$	$r(s, a_0)$
$a_1$	$r(s, a_1)$
$\vdots$	$\vdots$
$a_k^*$	$r(s, a_k^*)$
$\vdots$	$\vdots$
$a_n$	$r(s, a_n)$

Table 3.4: Agent  $j$ 's Convergent  $Q$ -value

The randomized algorithm  $\mathcal{M}$  in our case is Laplace mechanism introduced in **Definition 2.6**, Where the sensitivity  $S$  satisfies

$$S = \max_{f(Q_i(s)), f(Q_j(s))} \|f(Q_i(s)) - f(Q_j(s))\|_1$$

### 3.5.2.3 Euclidean Distance

We use Euclidean Distance to calculate the difference of  $Q$ -value between teacher and student's. Euclidean distance is most commonly used to calculate the distance between two vectors in a Euclidean space due to its simplicity. Let  $x, y$  be two  $N$ -dimension vectors and  $x = (x^1, x^2, \dots, x^N)$ ,  $y = (y^1, y^2, \dots, y^N)$ . The Euclidean distance  $d_E^2(x, y)$  is given by

$$(3.1) \quad d_E^2(x, y) = \sum_{i=1}^N (x^i - y^i)^2$$

We set a threshold  $d$  that when the distance  $d_E^2(\text{teacher}, \text{student}) > d$ , we consider the teacher to be sufficiently more experienced than the student to be able to provide advice. We have presented the convergence proof of  $Q(s, a)$  in our setting in the above subsection.  $Q(s, a)$  is convergent to  $r(s, a)$ , so two agents can learn the same knowledge when they are in the completely same setting.  $d_E^2(\text{teacher}, \text{student})$  is the knowledge gap that indicates how much more the teacher has learned compared to the student.

## 3.5.3 Algorithm Of Differentially Private Advising Framework

The **Algorithm 5** is the algorithm of our proposed differentially private advising framework. In this algorithm, we assume the agent  $j$  as a student and the agent  $i$  as a

**Algorithm 5** Differentially Private Advising Framework

---

**Require:** /\* Taking student agent  $j$  and teacher agent  $i$  at state  $s$  as an example \*/

**Require:** Initialize probability distribution,  $Q$ -value for each actions

**Require:**  $b_{give} = c$  and  $b_{ask} = c$  for each agent,  $c$  is a constant

**Require:** Sensitivity  $S$  and privacy budget  $\epsilon$

**Require:** The distance threshold  $d$

- 1: **function** EUCLIDEAN DISTANCE( $i, j$ )
- 2:   **if**  $\mathcal{A}(i) == \mathcal{A}(j)$  **then**
- 3:      $d_E^2(i, j) = \sum_{p=1}^{len(\mathcal{A}^j)} (Q_i(s, a_p) - Q_j(s, a_p))^2$
- 4:   **else if**  $len(\mathcal{A}(i)) > len(\mathcal{A}(j))$  **then**
- 5:      $d_E^2(i, j) = \sum_{p=1}^{len(\mathcal{A}^j)} [(Q_i(s, a_p) + Laplace(\frac{S}{\epsilon})) - Q_j(s, a_p)]^2$
- 6:   **else if**  $len(\mathcal{A}(i)) < len(\mathcal{A}(j))$  **then**
- 7:      $a_{diff}$ : the action **in**  $\mathcal{A}(j)$  but **not in**  $\mathcal{A}(i)$
- 8:      $d_E^2(i, j) = \sum_{p=1}^{len(\mathcal{A}^j)} [(Q_i(s, a_p) + Laplace(\frac{S}{\epsilon})) - Q_j(s, a_p)]^2 + (Q_j(s, a_{diff}) + Laplace(\frac{S}{\epsilon}))^2$
- 9:   **else if**  $len(\mathcal{A}(i)) == len(\mathcal{A}(j))$  and  $\mathcal{A}(i) \neq \mathcal{A}(j)$  **then**
- 10:      $a_{diff1}$ : the action **in**  $\mathcal{A}(j)$  but **not in**  $\mathcal{A}(i)$
- 11:      $a_{diff2}$ : the action **in**  $\mathcal{A}(i)$  but **not in**  $\mathcal{A}(j)$
- 12:      $d_E^2(i, j) = \sum_{p=1}^{len(\mathcal{A}^j)-1} [(Q_i(s, a_p) + Laplace(\frac{S}{\epsilon})) - Q_j(s, a_p)]^2 + [(Q_i(s, a_{diff2}) + Laplace(\frac{S}{\epsilon})) - Q_j(s, a_{diff1})]^2$
- 13:   **end if**
- 14:   **return**  $d_E^2(i, j)$
- 15: **end function**
- 16: **repeat** for each epoch
- 17:   **if**  $b_{ask}(j) > 0$  **then**
- 18:      $i = [argmax_{i \in |Neig|} (EuclideanDistance(i, j)) \text{ and } b_{give}(i) > 0]$
- 19:     **if**  $d_E^2(i, j) > d$  **then**
- 20:        $\bar{r} \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(s, a) (Q_i(s, a) + Laplace(\frac{S}{\epsilon}))$ ;
- 21:       **for**  $a \in \mathcal{A}_j(s)$  **do**
- 22:          $\pi'(s, a) \leftarrow \pi(s, a) + \xi(Q_i(s, a) + Laplace(\frac{S}{\epsilon}))$ ;
- 23:       **end for**
- 24:        $i$  selects an action  $a_k$  for  $j$ , based on the new probability distribution:  
 $\pi'(s) = \langle \pi'(s, a_1), \dots, \pi'(s, a_n) \rangle$ ;
- 25:        $r \leftarrow \mathcal{R}(s, a_k)$ ;
- 26:        $Q(s, a_k) \leftarrow (1 - \alpha)Q(s, a_k) + \alpha [r + \gamma \max_a Q(s', a)]$ ;
- 27:        $\bar{r} \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(s, a) Q(s, a)$ ;
- 28:       **for**  $a \in \mathcal{A}(s)$  **do**
- 29:          $\pi(s, a) \leftarrow \pi(s, a) + \xi(Q(s, a) - \bar{r})$ ;
- 30:       **end for**
- 31:        $\pi(s) \leftarrow \text{Algorithm 2}(\pi(s))$ ;
- 32:        $s \leftarrow s'$ ;
- 33:        $b_{ask}(j) - 1$ ;
- 34:        $b_{give}(i) - 1$
- 35:     **else Algorithm 1**
- 36:     **end if**
- 37:   **else Algorithm 1**
- 38:   **end if**
- 39: **until** finish learning

---

teacher, and set the distance threshold  $d$  to trigger the advising process. We include Euclidean distance, DP mechanism and teacher-student advising framework into the RL algorithm. We firstly propose the function Euclidean distance with DP to decide the timing of advising. There are four possible situations involved in our scenario to calculate Euclidean distance between  $i$  and  $j$ 's  $Q$ -value. Firstly, when two agents have identical actions, they directly calculate Euclidean distance based on their original  $Q(s)$ . In the second situation, when the teacher has one action more than the student, we add *Laplace* noise onto the teacher's  $Q(s)$  and only calculate the Euclidean distance of same actions'  $Q$ -value between teacher and student. Thirdly, when the teacher has one action fewer than the student, we temporarily add the missing action to the teacher with an initial value of  $Q$ -value, normally 0. Following this, we add *Laplace* noise onto the teacher's new  $Q$ -value including the additional action to calculate Euclidean distance with the student's  $Q(s)$ . The last situation is when the teacher and student have the same number of actions but one of them is different. In this situation, we temporarily consider the teacher's different action as the student's action differing with the teacher, and then add Laplace noise on the teacher's new  $Q(s)$  to calculate Euclidean distance with the student's  $Q(s)$  normally. All of these four situations satisfy the definition of neighbouring dataset, and are allowed within our proposed framework. This is how we calculate Euclidean distance to determine when advising happens.

We introduce the function  $EuclideanDistance(i, j)$  in the main part our algorithm to determine who the teacher agent is. While the agent  $j$  has not depleted its ask budget  $b_{ask}$ , it can ask advice from its 'neighbouring agents'. In the line 18,  $|Neig|$  is the dataset of agent  $j$ 's neighbouring agents who can provide advice to  $j$ . The agent  $j$  first checks its neighbouring agents to determine whether they still have remaining budget  $b_{give}$ , and then calculates the Euclidean distance with every neighbour with non-zero  $b_{give}$ . The teacher  $i$  must satisfy the following three requirements: 1)  $i$  does not deplete its  $b_{give}$ , 2) the Euclidean distance between  $i$  and  $j$  is the greatest among all neighbouring agents, and 3) their Euclidean distance is greater than the threshold  $d$ . If existing an agent  $i$  satisfies these three requirements, it can select an action based on its new probability distribution  $\pi'(s)$  transitioned from a new  $Q(s)$  during the function  $EuclideanDistance(i, j)$  (line 20 to 24). Remarkably, the  $i$ 's noised  $Q$ -value is only involved in the Euclidean distance calculation and selecting action for the student, but does not update  $i$ 's original  $Q$ -value and affect its learning process. The student agent  $j$  then gets the reward from this action and then adjusts the  $Q(s, a)$  and probability distribution. The advising process costs 1 for student's budget  $b_{ask}$  and teacher's budget

$b_{give}$ . If no neighbouring agent meets above three requirements, or the agent  $j$  runs out its asking budget  $b_{ask}$ , the agent  $j$  then proceeds with traditional RL as in **Algorithm 3**.

In general, our algorithm allows more advising processes to occur while guaranteeing the original teacher-student advising framework runs normally in an RL system. We use the DP mechanism to eliminate the difference between two agents' actions in a reasonable bound, which facilitates more effective advising with limited communication budgets.

## 3.6 Experiments

### 3.6.1 Experiment Setting

In order to demonstrate our method intuitively, we have designed experiments with the scenario introduced in Section 2. There are three parts of experiment. In the first experiment, we investigate the following three strategies:

1. **Traditional Reinforcement learning (No Advice)** – As reference, we evaluate the SARSA learning algorithm without advising;
2. **Traditional Advising Framework** – As reference, we evaluate the traditional teacher-student advising framework. Advising only happens between two agents with completely the same actions.
3. **Differentially Private Advising Framework** – Our proposed method. Advising process occurs between two agents with either the same actions or differing by one. This strategy applies DP, which means that students accept teachers knowledge with additional noise to adjust their own knowledge influencing action

We involve our approach and two compared strategies together in the experiment 1 via the controlled variable method where each strategy has the same number of users, servers and tasks. We also set the same learning rate, rewards and initial  $Q$ -value for each action to guarantee the total knowledge is the same among every strategy. The only different setting in our proposed method is that each agent has one different action, (i.e.  $\mathcal{A}(a_1) : \{A, B, C, D\}, \mathcal{A}(a_2) : \{A, B, C, E\}, \mathcal{A}(a_3) : \{A, B, D, E\}, \dots$ ), but each agent's actions are the same for the other two strategies. The purpose of this experiment is to demonstrate that our method can allow advising to occur between two agents with one different action. The expectation is that our algorithm can have a same performance with traditional advising framework.

Our second experiment is designed to demonstrate the second contribution that our proposed approach can widen the traditional advising framework. We compare two approaches' performance in a completely same setting. In this RL system, only a subset of agents have the same available actions and others have one different actions among each other. In this case, only a small amount of agents with the same actions can advise each other in traditional advising framework, but all of agents can advise their neighbouring agents in our proposed method. In theory, our method can trigger more advising processes to positively accelerate convergence. We use the traditional advising framework as the base to demonstrate how much performance our differentially private advising framework can improve. In the third experiment, we explore how the privacy budget  $\epsilon$  affect the DP performance in our algorithm. We set the epsilon as 0.1, 0.4, 0.7, 1 and 2 separately.

The comparison criteria is convergence ratio, the accuracy of agents' learning, which is described by the following equation:

$$\frac{\sum_{k \in \mathcal{A}}^n \pi(s, a_k) * \mathcal{R}(s, a_k)}{\sum_{m \in \mathcal{A}}^n \pi(s, a_m) * \mathcal{R}(s, a_m)}$$

where  $n$  is each agent,  $k$  is each action of an agent,  $\pi(s, a_k)$  is the probability of this action,  $\mathcal{R}(s, a_k)$  is the reward of this action,  $m$  is the action with theoretically maximum reward. Due to the fixed reward  $r(s, a)$ , the change of the ratio is affected by the change of probability. The greater convergence ratio means that agents have the higher probability to select the action with higher reward.

**TABLE 3.5** shows three experiments' parameters. Note that, we set the reward for the optimal action as 2 while others are 0. The Euclidean distance threshold is set to 0.7, calculated by the theoretical convergent value of  $Q$ -value regarding related settings. We assign an advising budget of 30 for both giving and asking for each agent during learning processes until convergence.

### 3.6.2 Experiment Results

The **Figure 3.4** shows the result of experiment 1 with the comparison between three strategies with the different setting of actions but in the same numbers. Our proposed approach is shown with the red line, and other two approaches are demonstrated with blue and black lines separately. The result shows that all three strategies are convergent to about 92%, and the red line is above the other two, that is, our proposed method converges the fastest, especially in the early learning stage from 0 to 250 epochs. The



Experiment		Experiment 1	Experiment 2	Experiment 3
		Parameters		
users	$n_{users}$	10		
servers	$n_{servers}$	12		
tasks	$n_{tasks}$	10		
privacy budget	$\epsilon$	1	1	(0, 2)
privacy sensitivity	$S$	1		
reward	$r$	2, 0		
Euclidean distance threshold	$d$	0.7		
Budget for giving advice	$b_{give}$	30		
Budget for asking advice	$b_{ask}$	30		
epoch		700		

Table 3.5: Parameter setting

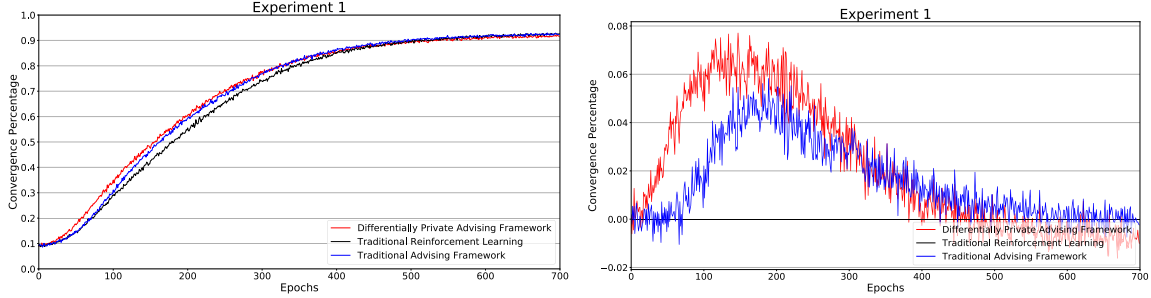


Figure 3.4: Experiment 1: Comparison between three strategies

Figure 3.5: Experiment 1: Performance Comparison Between three strategies (Traditional Reinforcement Learning as base)

details are shown in **Figure 3.5** which is the performance comparison between three strategies with the traditional RL as a base to check how much we improve the learning process. Although our proposed method may slightly drop the convergence rate within 0.5% after 400 epochs due to the excess advises in some epochs and its negative impact from the DP noise on agents' Q tables, our proposed method has a higher convergence percentage than the other two. Our proposed method makes a huge improvement which is around 7% from the beginning to 200 epochs, which means our approach can accelerate agents' learning during the whole learning process, especially in the early stage.

The experiment 2 demonstrates our second contribution that our approach allows more advising processes to occur in an RL system. **Figure 3.6** shows the result of experiment 2 with the comparison among three mentioned strategies in an identical environment, which differs from experiment 1 in that we only guarantee the total knowledge is the same for three strategies but with different actions in experiment 1.

CHAPTER 3. DIFFERENTIALLY PRIVATE ADVISING FRAMEWORK IN SIMULTANEOUSLY REINFORCEMENT LEARNING

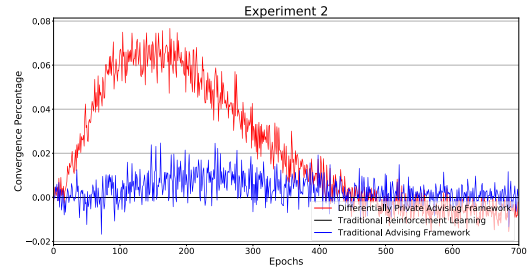
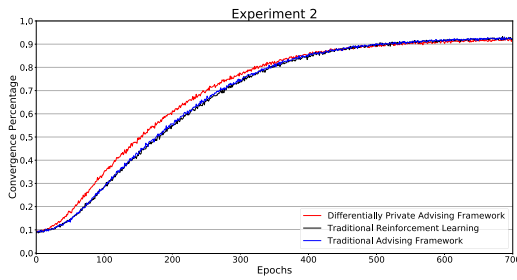


Figure 3.6: Experiment 2: Comparison between three strategies with the same set-  
 Figure 3.7: Experiment 2: Performance Comparison Between three strategies (Traditional Reinforcement Learning as base)

Similar to **Figure 3.5**, **Figure 3.7** shows how much we improve the learning speed in this RL setting. The performance of the traditional advising framework shown by the blue line is very close to the traditional RL strategy, that is, it can only achieve a limited acceleration of learning due to the fact that advising can only occur among few agents with the same actions in the setting. However, our approach shown by the red line, accelerates the learning speed much more in the same setting than the other two strategies.

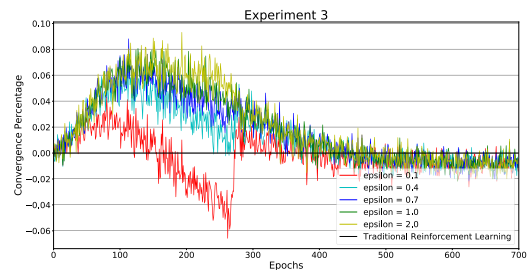
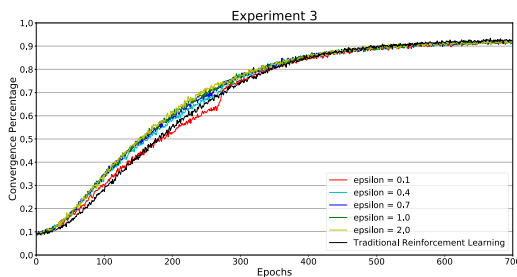


Figure 3.8: Experiment 3: Comparison between different epsilon value with the same setting  
 Figure 3.9: Experiment 3: Performance Comparison Between different epsilon value (Traditional Reinforcement Learning as base)

The experiment 3 explores the performance of DP with different value of epsilon. The experimental results shown in **Figure 3.8** and **Figure 3.9** indicate that the performance is increasingly higher with the greater value of epsilon. When  $\epsilon = 0.1$ , it even has a negative impact on the performance and performs worse than the traditional RL due to the impact from the large noise on agents' Q tables resulting in inefficient advises among agents.

### 3.6.3 Result Discussion

In this section, we will discuss and analyze our experimental results. Our purpose in the first experiment is to demonstrate our first contribution that agents can give advice to or ask advice from others even when they have different actions. Our original expectation for the proposed method was that it would have the same effect as the traditional advising framework. However, the experiment shows that we have a better performance in the early learning state with a higher convergent percentage. We believe that it is positively affected by DP noise. Referring back to our **Algorithm 5**, we see that a teacher calculates Euclidean distance with the noised  $Q$ -value with a student's  $Q$ -value, and transitions the noised  $Q$ -value to the new probability distribution to make the decision for the student. An appropriate randomized noise positively influences the calculation of Euclidean distance and facilitates more advising occurring in the early learning stage. The noise also keeps the shape of the probability distribution of the teacher, that is, the teacher still has a high probability to choose the best action, and this positively increases the result of Euclidean distance to easily satisfy the threshold to trigger advising process.

The experiment 2 corresponds to our second contribution that our method can widen the applicable field of advising in an RL system. Our approach performs much better than the other two compared strategies because our method allows more advising happen by breaking the limitation of the traditional advising framework. As we mentioned earlier, in the environment of this experiment, only a subset of agents have the same actions, and others have one different action with each other. The traditional advising framework performs worse because only a small number of agents with same actions can advise each other, and the effectiveness of advising is not obvious due to limited times that advising is able to occur and the effect from learning rate. However, our proposed method allows more potential advising to occur among agents with the same actions or neighbouring agents with one different action. Agents have more opportunities to ask advice from more experienced neighbouring agents. The learning process is positively affected by more advice and better advice in our framework. Therefore, our proposed method can widen the applicable field of advising by allowing more potential advising.

The experiment 3 demonstrates the performance of DP in our proposed framework. We set five different values of privacy budget  $\epsilon$  ranging from 0.1 to 2, to explore how the DP noise affects the performance. Referring back to **Equation 2.4** we can see that the scale of the randomized noise is determined by sensitivity  $S$  and privacy budget  $\epsilon$ , and we have mentioned that in our case  $S$  is fixed as 1 because the maximum value of a percentage is 1. So  $\epsilon$  is the only factor affecting the scale of noise with an inversely

proportional relationship. When  $\epsilon = 0.1$ , our proposed method even performs worse than traditional RL because the noise is so large that it negatively impacts on the teacher’s selection and student’s action. It is obvious that the red line has a sharp increase at epoch 300 because most agents have run out their asking and giving budgets and start independently learning without the negative impact from the randomized noise. Therefore, in **Figure 3.8** and **Figure 3.9**, it is clear that our algorithm perform better when the value of  $\epsilon$  is greater within a reasonable range.

### 3.7 Summary and Future Work

In this chapter, we introduce a novel differentially private advising framework for MARL applying into a real-world cloud server environment. Our proposed method allows agents with at most one different action to advise each other to accelerate learning speed toward convergence in a simultaneously learning setting. We applied DP mechanism to add a randomized noise on the teacher agent’s  $Q$ -value, and calculated the Euclidean distance between teacher’s noised  $Q$ -value and student’s  $Q$ -value to decide the timing of advising. If the Euclidean distance is greater than a threshold, the teacher agent selects an action for students based on the new probability distribution transitioned from the new noised  $Q$ -value.

We summarize three key contributions of our chapter: 1) we break the limitation of conventional advising framework which only allows advising between agents with the same available actions, 2) we widen the applicable field of conventional advising framework and thus allow more potential advising in some RL systems, and 3) we design a new approach to decide when advising occurs by calculating the Euclidean distance of teacher and student’s  $Q$ -value.

Our three experimental results demonstrate our contributions and also explore how the DP budget  $\epsilon$  affects the DP in the proposed method. The experimental results indicate that our framework can facilitate potential advising between two agents with one different action, and can perform better in the early learning state than the traditional advising framework in the same setting. Our approach may direct some further research in server selection, resource allocation and task allocation problems.

## DIFFERENTIALLY PRIVATE MULTI-AGENT PLANNING FOR LOGISTIC-LIKE PROBLEMS

Planning is one of the main approaches used to improve agents' working efficiency in a MARL system by making plans beforehand. However, during planning, agents face the risk of having their private information leaked. This chapter proposes a novel strong privacy-preserving planning approach for logistic-like problems. This approach outperforms existing approaches by addressing two challenges: 1) simultaneously achieving strong privacy, completeness and efficiency, and 2) addressing communication constraints. These two challenges are prevalent in many real-world applications including logistics in military environments and packet routing in networks. To tackle these two challenges, our approach adopts the DP technique, which can both guarantee strong privacy and control communication overhead. To the best of our knowledge, this is the first work to apply DP to the field of multi-agent planning as a means of preserving the privacy of agents for logistic-like problems. We theoretically prove the strong privacy and completeness of our approach and empirically demonstrate its efficiency. We also theoretically analyze the communication overhead of our approach and illustrate how DP can be used to control it.

## 4.1 Introduction

Multi-agent planning is one of the fundamental research problems in multi-agent systems [26, 135] and has been widely used in real-world applications [71, 85]. Multi-agent planning research aims to improve agents' working efficiency by making plans in advance. In traditional multi-agent planning, each agent makes plans independently [24]. However, in complex problems, such as logistics, agents have to work together collaboratively to make plans in order to achieve joint goals [73]. Collaborative multi-agent planning is thus required. Collaborative multi-agent planning has been applied to various real-world applications, such as logistics [72], job machine assignment [59], and aircraft simulation [74]. Research into collaborative multi-agent planning largely focuses on jointly automated planning [118]. During jointly automated planning, agents have to share information. However, this kind of information sharing often results in the leaking of agents' private information. Accordingly, to protect agents' privacy, privacy preservation is introduced into the collaborative multi-agent planning process [101, 102]. The main problem associated with privacy preservation in collaborative multi-agent planning is that of how to make plans for agents while also preserving the privacy of each agent.

Privacy can be roughly classified into four levels: weak privacy, strong privacy, object cardinality privacy, and agent privacy [118]. Specifically, weak privacy means that an agent does not explicitly disclose its private information to others. Strong privacy means that an agent, regardless of its reasoning power, cannot deduce the private information of other agents based on the information available to it. Developing a planning method with strong privacy in distributed and communication-constrained environments is challenging for the following two reasons. First, it is difficult to achieve strong privacy, completeness and efficiency simultaneously [123]. Second, in communication-constrained environments, each agent is allowed to communicate only a limited number of times.

These two challenges are widespread in many real-world applications. A typical application is military logistics. In military logistics, it is vital that each military unit should strongly protect its private and sensitive facts. Also, plans for military units must be complete and efficient to avoid any delay. In addition, communication between units has to be constrained, since the more communication takes place, the more likely it will be that sensitive information is leaked. Due to the lack of information in such environments, It is very challenging to make strongly private and efficient plans with only limited view and communication.

Most existing planning approaches are either weak privacy-preserving or overlook

the issue of privacy preservation entirely [118]. Very few approaches are strong privacy-preserving [12]. These strong privacy-preserving planning approaches, however, may not achieve strong privacy, completeness and efficiency simultaneously, as summarized in [123]. Moreover, these approaches also may not work efficiently in distributed and communication-constrained environments, as they implicitly assume that an agent can communicate directly with all other agents, and overlook the analysis of communication overhead.

Accordingly, in this chapter, we develop a novel strong privacy-preserving planning approach for distributed and communication-constrained environments. Our approach focuses primarily on logistic-like problems, which are typically used as running examples in multi-agent planning. To achieve strong privacy, completeness and efficiency simultaneously, we adopt the DP technique. DP is a promising privacy model, which has been mathematically proven that when this model is in use, an individual record being stored in or removed from a dataset makes little difference to the analytical output of the dataset [22, 143]. To the best of our knowledge, we are the first to apply DP to the privacy-preserving planning problem. Using a DP mechanism to obfuscate an agent's private information can strongly preserve the agent's privacy while also having minimal impact on the usability of the agent's private information.

Furthermore, we also address the communication-constrained environment issue by adopting the concept of a 'privacy budget'. In DP, a privacy budget is applied to control privacy levels. In our proposed approach, the privacy budget can naturally be used to control communication overhead, with the result that only a limited number of messages are permitted during a planning phase. In summary, the contributions of this chapter are two-fold:

- Improving upon existing strong privacy-preserving planning approaches, our approach can achieve strong privacy, completeness and efficiency simultaneously in logistic-like problems using the DP technique.
- Our approach is more applicable to distributed and communication-constrained logistic-like problems than existing approaches.

The remainder of this chapter is organized as follows. In the next section, a detailed review of related work is presented. Then, a motivating example is given in Section 4.3. Preliminaries are presented in Section 4.4. After that, the novel planning approach and the theoretical analysis are presented in Sections 4.5 and 4.6, respectively. The application of our approach to other domains is illustrated in Section 4.7. Next, the

experimental results are provided in Section 4.8. Finally, Section 4.9 concludes this chapter.

## 4.2 Related Work

### 4.2.1 Weak privacy-preserving approaches

Torreno et al. [119] develop a framework known as FMAP (forward multi-agent planning). In FMAP, agents maintain a common open list with unexplored refinement plans. Agents then jointly select an unexplored refinement plan. Each agent then expands the plan using a forward-chaining procedure. Agents exchange these plans and use a distributed heuristic approach to evaluate them. Later, based on the FMAP framework, Torreno et al. [120] develop a set of global heuristic functions: DTG (domain transition graphs) heuristic and landmarks heuristic, in order to improve the efficiency of the FMAP framework.

Stolba and Komenda [109] present a multi-agent distributed and local asynchronous (MADLA) planner. This planner adopts a distributed state-space forward-chaining multi-heuristic search. The multi-heuristic search takes the advantages of both local and distributed heuristic searches by combining them together. As a result, the combination of the two heuristics outperforms the two heuristics separately.

Maliah et al. [73] propose a greedy privacy-preserving planner (GPPP). In GPPP, agents collaboratively generate an abstract global plan based on two privacy-preserving heuristics: landmark-based heuristic and privacy-preserving pattern database heuristic. Each agent generates a local plan by extending the global plan.

### 4.2.2 Strong privacy-preserving approaches

Brafman [12] is the first to theoretically prove strong privacy in multi-agent planning. He proposes an approach referred to as Secure-MAFS (secure multi-agent forward search). Secure-MAFS extends the MAFS approach [82] by reducing the amount of information exchanged between agents. In Secure-MAFS, agents protect their privacy by opting not to communicate a given two states to others if these two states differ only in their private elements. This is because other agents could possibly deduce private information through the non-private or public part of the states.

Tozicka et al. [123] investigate the limits of strong privacy-preserving planning. They formulate three aspects of strong privacy-preserving planning: privacy, completeness, and efficiency. They theoretically find that these three aspects are difficult to achieve



at the same time for a wide class of planning algorithms. Also, they develop a strong privacy-preserving planner that embodies a family of planning algorithms. The planner is based on private set intersection, which has been proven to be computationally secure.

Stolba et al. [110, 111] refine privacy metrics by quantifying the amount of privacy loss. In this case, their analysis of privacy loss is conducted by assessing information leakage [14, 107]. The amount of information leakage is measured as the difference between initial uncertainty and remaining uncertainty. They also develop a general approach to compute the privacy loss of search-based multi-agent planners. This computation is based on search tree reconstruction and classification of leaked information pertaining to the applicability of actions.

### 4.2.3 Other privacy-preserving approaches

Some other existing works seem to be related to ours, such as differentially private networks [35] and privacy-preserving distributed constraint optimization [138]. However, the research aims of these works differ from ours. The research of differentially private networks mainly aims at hiding specific information contained in a network, which may be disclosed by answering queries regarding that network. By contrast, multi-agent privacy-preserving planning aims at collaboratively making plans without revealing the private facts of each participating agent. In [56], Kasiviswanathan et al. develop a set of node-differentially private algorithms to engage in the private analysis of network data. The key concept here is to obfuscate the input graph onto the set of graphs with maximum degree below a certain threshold. Blocki et al. [10] improve accuracy in differentially private data analysis by introducing the notion of restricted sensitivity in order to reduce noise. Restricted sensitivity represents the sensitivity of a query only over a specific subset of all possible networks.

Proserpio et al. [94] propose a platform for differentially private data analysis: wPINQ (weighted Privacy Integrated Query). wPINQ treats edges as a weighted dataset on which it performs  $\epsilon$ -differentially private computations, such as manipulation of records and their weights. Thus, the presence or absence of individual edges can be masked. Fioretto et al. [35] design a privacy-preserving obfuscation mechanism for critical infrastructure networks. Their mechanism consists of three phases: 1) obfuscating the locations of nodes using the exponential mechanism, 2) obfuscating the values of nodes using the Laplace mechanism, and 3) redistributing the noise introduced in the previous two phases using a bi-level optimization problem. These works assume the existence of adversaries while in multi-agent planning, agents are typically assumed to be honest but curious.

Research into privacy-preserving distributed constraint optimization aims at securely coordinating the value assignment for the variables under a set of constraints in order to optimize a global objective function [36]. By contrast, multi-agent privacy-preserving planning aims at securely making plans that enable individual agents to achieve their goals. Grinshpoun and Tassa [45] devise a novel distributed constraint optimization problem (DCOP) algorithm that preserves constraint privacy. In their problem, a group of agents needs to compare the sum of private inputs possessed by those agents against an upper bound held by another agent. During this comparison, none of these agents learns information on either the sum or the private inputs of other agents. Their algorithm accomplishes this through the use of a secure summation protocol and a secure comparison protocol.

Tassa et al. [115] propose a DCOP algorithm that is immune to collusion and offers constraint, topology and decision privacy. To achieve this goal, they adopt a secure multi-party computation protocol [9] which is capable of securely comparing the cost of the current full assignment and the upper bound and guaranteeing the security of collusion of up to half of the total agents. From an examination of the two above-mentioned works, it can be seen that the privacy-preserving DCOP mainly focuses on securely comparing the values of variables against an upper bound, while multi-agent privacy-preserving planning mainly focuses on the secure computation of each individual agent.

### 4.3 A Motivating Example

**Figure 4.1** presents a military logistic map. In this map, a circle denotes a military base while a rectangle denotes a logistic center. The lines connecting the bases and logistic centers are routes. Each route has a length, which is not indicated on the map in the interests of clarity. Each letter in a circle indicates a military unit's name, while each number in a circle is the index of a base in the military unit's local area. For example, ' $(a, 3)$ ' denotes the third base in military unit  $a$ 's local area. Six military units are included on this map:  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$ , and  $f$ . Each unit exclusively operates in a local area of the map.

Information about a local area is private to the corresponding military unit. This information includes 1) the number of military bases in this local area, 2) the number of routes in this local area, 3) the length of these routes in this local area, and 4) the positions of packages in this local area. However, information regarding whether a given package is or is not located in a particular logistic center is public. For example, in **Figure 4.2**, we extract military unit  $a$ 's local area from **Figure 4.1**. In **Figure 4.2**, there

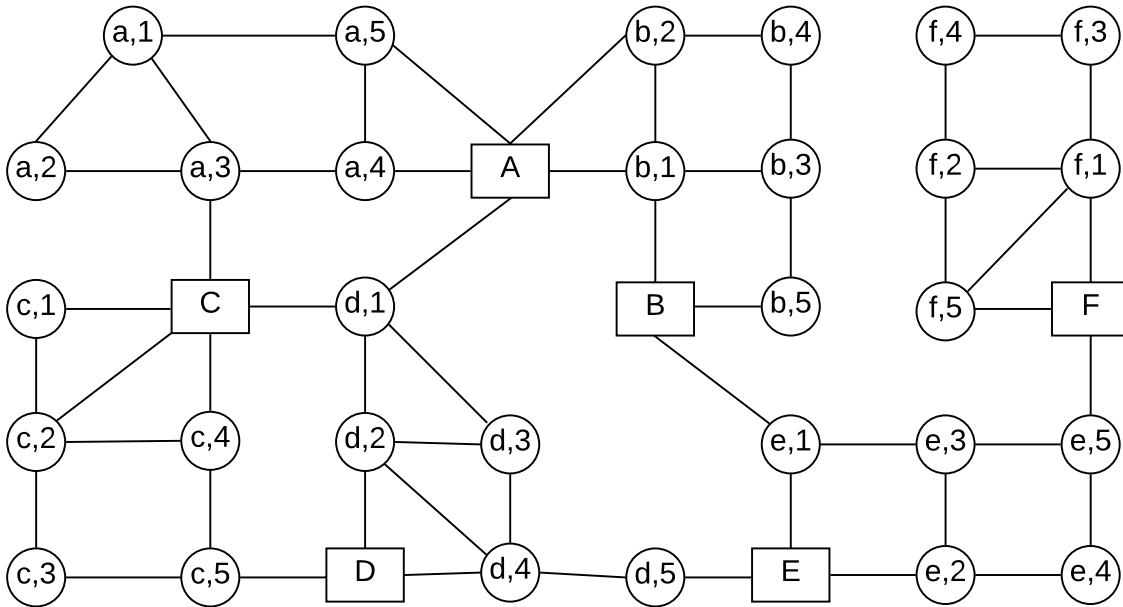


Figure 4.1: An example of a logistic map

are five bases:  $(a, 1)$ ,  $(a, 2)$ ,  $(a, 3)$ ,  $(a, 4)$  and  $(a, 5)$ . The number of these bases and routes is private to military unit  $a$ . Moreover, the length of these routes is also private to unit  $a$ . As noted above, the information that a package is located in logistic center  $A$  is public and known to all military units.

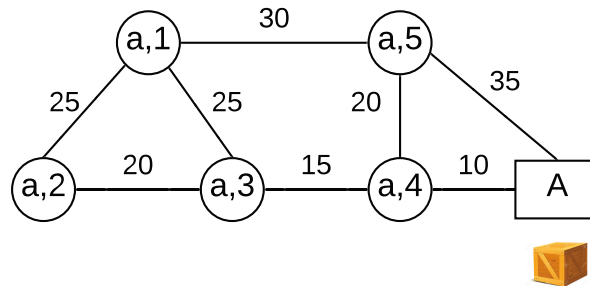


Figure 4.2: Unit  $a$ 's local area

The problem in this example is as follows: how should a plan be made for a military unit to transport a package from one base to another, while strongly preserving each military unit's privacy? For example, unit  $a$  wants to transport a package from  $(a, 2)$  to  $(f, 4)$ , but  $(f, 4)$  is located in military unit  $f$ 's local area. Thus, multiple units must collaborate to make a plan to deliver the package, while each unit's privacy is required to be strongly preserved during this process. This problem therefore includes the above-mentioned two challenges. First, planning for military units is highly expected to achieve

strong privacy, completeness and efficiency simultaneously, especially when military units are involved in a war. Second, the communication of each military unit may be constrained, as increased level of communication may result in a higher chance of private information being leaked [83]. As the above two challenges have not been adequately addressed by existing approaches, these approaches may not be suitable for this environment. Accordingly, in this chapter, a novel strong privacy-preserving planning approach is proposed that takes these two challenges into account.

## 4.4 Preliminaries

### 4.4.1 The planning model

We propose a multi-agent planning model, Graph-STRIPS, which is based on a widely used privacy-aware planning model, MA-STRIPS [13]. Graph-STRIPS is defined by a 12-tuple:  $\langle \mathcal{AG}, \mathcal{V}, \{\mathcal{V}_i\}_{i=1}^m, \mathcal{V}_{Pub}, \mathcal{E}, \{\mathcal{E}_i\}_{i=1}^m, \mathcal{P}, \{\mathcal{P}_i\}_{i=1}^m, \mathcal{A}, \{\mathcal{A}_i\}_{i=1}^m, \mathcal{I}, \mathcal{G} \rangle$ :

- $\mathcal{AG}$  is a set of agents in the environment;
- $\mathcal{V}$  is a set of nodes (e.g., physical entities) in the environment;
- $\mathcal{V}_i$  is the set of nodes private to agent  $i$ ;
- $\mathcal{V}_{Pub}$  is the set of public nodes in the environment,  $\mathcal{V}_{Pub} = \mathcal{V} - \bigcup_{l=1}^{|\mathcal{AG}|} \mathcal{V}_l$ ;
- $\mathcal{E}$  is a set of edges (e.g., the relationships between physical entities) in the environment;
- $\mathcal{E}_i$  is the set of edges private to agent  $i$ ;
- $\mathcal{P}$  is a set of possible facts about the environment;
- $\mathcal{P}_i$  is the set of private facts of agent  $i$ ;
- $\mathcal{A}$  is a set of possible actions of all the agents;
- $\mathcal{A}_i$  is the set of private actions of agent  $i$ ;
- $m$  is the number of agents in the environment;
- $\mathcal{I}$  is the initial state of the environment;
- $\mathcal{G}$  is the goal state.

For example, in **Figure 4.1**, each military unit is modelled as an agent. In this case, we have the following:

- $\mathcal{AG} = \{a, b, c, d, e, f\}$  and  $m = 6$ ;
- $\mathcal{V}$  is the set of military bases and logistic centers;
- $\mathcal{V}_i$  denotes the set of bases in the local area of agent  $i$ ; for example, in agent  $a$ 's local area,  $\mathcal{V}_a = \{(a, 1), (a, 2), (a, 3), (a, 4), (a, 5)\}$ ;
- $\mathcal{V}_{Pub}$  denotes the set of logistic centers;
- $\mathcal{E}$  is the set of routes connecting bases and centers;
- $\mathcal{E}_i$  denotes the set of routes in the local area of agent  $i$ ; for example, in agent  $a$ 's local area,  $\mathcal{E}_a = \{(a, 1) \sim (a, 2), (a, 2) \sim (a, 3), (a, 3) \sim (a, 4), \dots\}$ ;
- $\mathcal{P}$  includes the position of bases, logistic centers and packages;
- $\mathcal{P}_i$  includes 1) the position of packages in the local area of agent  $i$ ; for example, if agent  $a$  has a package in  $(a, 1)$ , then  $\mathcal{P}_a = \{package\_in\_ (a, 1)\}$ ; 2) the number of bases in the local area of agent  $i$ ; 3) the number of routes in the local area of agent  $i$  and 4) the length of these routes.
- $\mathcal{A}$  includes the actions of moving from a base or a logistic center to another base or logistic center;
- $\mathcal{A}_i$  includes the actions of moving from a base or a logistic center to another base or logistic center in the local area of agent  $i$ ; for example, an action of agent  $a$  can be: *moving from  $(a, 1)$  to  $(a, 2)$*  which is abbreviated as  $(a, 1) \rightarrow (a, 2)$ , where the pre-condition of this action is  $package\_in\_ (a, 1)$  and the effect of this action is  $package\_in\_ (a, 2)$ ;
- If  $a$  wants to transport a package from  $(a, 3)$  to  $(e, 2)$ , then  $\mathcal{S} = \{package\_in\_ (a, 3)\}$  and  $\mathcal{G} = \{package\_in\_ (e, 2)\}$ ;  $\mathcal{V}_{\mathcal{S}} = (a, 3)$  and  $\mathcal{V}_{\mathcal{G}} = (e, 2)$ .

If agent  $a$  is to transport a package from  $(a, 3)$  to  $(e, 2)$ , the associated plan could be  $\Pi_a^{\triangleright} = \langle \mathcal{V}_{\mathcal{S}} \rightarrow (a, 4), (a, 4) \rightarrow A, A \rightarrow B, B \rightarrow E, E \rightarrow \mathcal{V}_{\mathcal{G}} \rangle$ . In plan  $\Pi_a^{\triangleright}$ , the details of how to move from  $A$  to  $B$ , from  $B$  to  $E$  and from  $E$  to  $(e, 2)$  are not included, as these details involve other agents' private information that is unknown to agent  $a$ . In fact, as  $(e, 2)$  is

private to agent  $e$ , agent  $a$  is unaware of the existence of  $(e, 2)$ . Agent  $a$ , however, knows that the destination is in agent  $e$ 's local area.

Specifically, each agent's private information includes two parts: private facts and private actions. An agent's private facts include four components: 1) the number of nodes in its local area, i.e., the number of military bases in the logistic example, 2) the number of edges in its local area, i.e., the number of routes in the logistic example, 3) the length of these edges, i.e., the length of routes in the logistic example and 4) the positions of any items in its local area, i.e., the positions of packages in the logistic example. An agent's private actions are the movements of items in its local area. In this private information, the positions and movements of items are not required by other agents. Thus, these two pieces of information will not be disclosed to other agents. For the other three pieces of information: the number of nodes, the number of edges and the length of edges, since agents have to share the three pieces of information for planning, we need to develop a privacy-preserving mechanism to protect them.

Formally, we have the following definition.

**Definition 4.1** (Agents' privacy). An agent  $i$ 's privacy is defined as a 3-tuple:  $\langle \mathcal{V}_i, \mathcal{E}_i, L(\mathcal{E}_i) \rangle$ , where  $\mathcal{V}_i$  is the set of nodes in agent  $i$ 's local area,  $\mathcal{E}_i$  is the set of edges and  $L(\mathcal{E}_i)$  denotes the set of length of the edges.

To protect the privacy of  $\mathcal{V}_i$  and  $\mathcal{E}_i$ , we adopt the node-DP technique and uses the Laplace mechanism to mask the number of both nodes and edges. To protect the privacy of  $L(\mathcal{E}_i)$ , we adopt the exponential mechanism along with an RL algorithm.

#### 4.4.2 Privacy-preserving multi-agent planning

Privacy-preserving multi-agent planning aims to prevent private facts and private actions from being revealed. The idea behind privacy-preserving multi-agent planning is based mainly on research in the field of secure multi-party computation [140], where multiple agents jointly compute a function while each agent possesses private input data. The goal is to compute the function without revealing agents' private input data.

One intuitive solution would be to simply not disclose any private information to others. However, since an agent must collaborate with other agents in order to achieve its goals, it is infeasible to hide all private information completely. To ensure that this private information is disclosed securely to the other agents, it is necessary to use privacy-preserving techniques.

**Definition 4.2** (Strong Privacy [118]). A multi-agent planning approach is strong privacy-preserving if none of the agents is able to infer any private facts regarding an agent’s tasks from the public information it obtains during planning. A planning approach is strong privacy-preserving if agents cannot deduce extra private information based on public information and exchanged modified private information.

Generally speaking, the goal of strong privacy is to preserve all private information of each agent. More specifically, in the logistics example in Section 4.3, an agent’s private information includes 1) the topology of an agent’s local map, 2) the length of each route on an agent’s local map, and 3) an agent’s private facts and private actions as defined in Section 4.4.1. To guarantee strong privacy, it is necessary to consider several factors, such as the nature of the communication channel or the computational power of the agents [118].

We adopt DP to achieve strong privacy. In addition to a privacy guarantee, a planning approach also needs soundness and completeness guarantees.

**Definition 4.3** (Soundness [73]). A planning approach is sound if and only if (iff), for a given task, there is at least one valid plan followed by all participating agents to reach the goal state.

**Definition 4.4** (Completeness [123]). A planning approach is complete iff, for a given task, 1) the approach is sound and 2) the approach can guarantee to create a valid plan.

If a graph is treated as a dataset, a given node in the graph can be interpreted as a record in the dataset. According to Definition 2.2, we can have a similar definition for  $\epsilon$ -node-DP as follows.

**Definition 4.5** ( $\epsilon$ -node-DP [96]). A mechanism  $\mathcal{M}$  gives  $\epsilon$ -node-DP for any input pair of neighboring graphs  $G$  and  $G'$ , where  $G$  and  $G'$  differ by at most one node, and for any possible output set,  $\Omega$ , if  $\mathcal{M}$  satisfies:

$$(4.1) \quad Pr[\mathcal{M}(G) \in \Omega] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(G') \in \Omega]$$

Node-DP guarantees similar output distributions on any pair of neighboring graphs that differ in one node and the edges adjacent to that node. Thus, the privacy of both nodes and edges can be preserved.

## 4.5 The strong privacy-preserving planning approach

In this section, we first outline our approach in a general form, then use the aforementioned logistic example to instantiate our approach. A generalized form of our approach is presented in **Algorithm 6**. In Line 5 of **Algorithm 6**, agent  $i$  takes all the available public nodes into account to create a plan. These available public nodes are on the way from the initial state to the goal state and found by agent  $i$  during its searching phase. However, some of these available public nodes are not needed in the final plan. Then, in Line 8, agent  $i$  uses an RL algorithm to find the shortest route from the initial state to the goal state, and selects the public nodes on the shortest route to create a plan. The learning is based on the information obtained in Lines 6 and 7.

---

**Algorithm 6** The general form of our approach

---

- 1: /\*Take agent  $i \in \mathcal{AG}$  as an example;\*/
  - 2: **Input**: agent  $i$ 's local sets:  $\mathcal{V}_i, \mathcal{E}_i, \mathcal{P}_i, \mathcal{A}_i$ , and all the public facts and actions; also, the initial state  $\mathcal{S}$  and the goal state  $\mathcal{G}$ ;
  - 3: **Output**: a complete plan  $\Pi_i^\triangleright$  from  $\mathcal{S}$  to  $\mathcal{G}$ ;
  - 4: Agent  $i$  identifies  $\mathcal{V}_\mathcal{S}$  and  $\mathcal{V}_\mathcal{G}$  from the initial state  $\mathcal{S}$  and the goal state  $\mathcal{G}$ , respectively, and initializes plan:  $\Pi_i^\triangleright = \langle \mathcal{V}_\mathcal{S} \rightarrow \mathcal{V}_\mathcal{G} \rangle$ ;
  - 5: Agent  $i$  searches the goal state, and details plan  $\Pi_i^\triangleright$  by adding the available public actions into plan  $\Pi_i^\triangleright$ :  $\Pi_i^\triangleright = \langle \mathcal{V}_\mathcal{S} \rightarrow v_j, \dots, v_k \rightarrow \mathcal{V}_\mathcal{G} \rangle$ , where  $\{v_j, \dots, v_k\} \subset \mathcal{V}_{Pub}$ ;
  - 6: Agent  $i$  queries the intermediate agents to request local private facts;
  - 7: Each of these intermediate agents obfuscates its local private facts using the DP technique;
  - 8: Agent  $i$  uses the obfuscated facts to refine the plan by removing unnecessary public actions by means of an RL algorithm:  $\Pi_i^\triangleright = \langle \mathcal{V}_\mathcal{S} \rightarrow v_x, \dots, v_y \rightarrow \mathcal{V}_\mathcal{G} \rangle$ , where  $j \leq x, y \leq k$ ;
  - 9: Each action in plan  $\Pi_i^\triangleright$  is further refined by each agent creating a local plan; for example, action  $\mathcal{V}_\mathcal{S} \rightarrow v_x$  is refined by agent  $i$  creating a local plan as  $\langle \mathcal{V}_\mathcal{S} \rightarrow v_{i_a}, \dots, v_{i_b} \rightarrow v_x \rangle$ , where  $\{v_{i_a}, \dots, v_{i_b}\} \subset \mathcal{V}_i$ ;
  - 10: Agent  $i$  merges these local plans to form a complete plan:  $\Pi_i^\triangleright = \langle \mathcal{V}_\mathcal{S} \rightarrow v_{i_a}, \dots, v_{i_b} \rightarrow v_x, \dots, v_y \rightarrow \mathcal{V}_\mathcal{G} \rangle$ ; note that the details of local plans, created by intermediate agents, are not shown in plan  $\Pi_i^\triangleright$ , since they contain non-obfuscated private facts belonging to the intermediate agents;
- 

To instantiate this general approach, we use the logistic example given in Section 4.3. In this example, we assume that 1) all routes in the logistic map are bi-directional; 2) each individual agent controls only one local area; and 3) there are no isolated nodes on the map. An agent follows three steps to create a plan:

- Step 1: the agent creates a high-level logistic map;



- Step 2: the agent asks the agents in the intermediate areas to provide route and map information;
- Step 3: the agent uses the received information to create a complete plan.

#### 4.5.1 Step 1: Creating a high-level map

In **Figure 4.1**, it is supposed that agent  $a$  has a package to transport from  $(a, 2)$  to  $(f, 4)$ . As agent  $f$  is not agent  $a$ 's neighbor,  $a$  must query its neighbors,  $b$ ,  $c$ , and  $d$ , regarding the position of  $f$ . Two agents are deemed neighbors if there is at least one logistic center connecting two military bases, such that one of these bases belongs to each of the agents. In the case that agents,  $b$ ,  $c$  and  $d$ , also do not have  $f$  as a neighbor, they pass this query on to their neighbors, e.g., agent  $e$ . Finally, agent  $f$  is found through agent  $e$ . By using the information acquired while finding agent  $f$ , agent  $a$  can create a high-level logistic map, as shown in **Figure 4.3**.

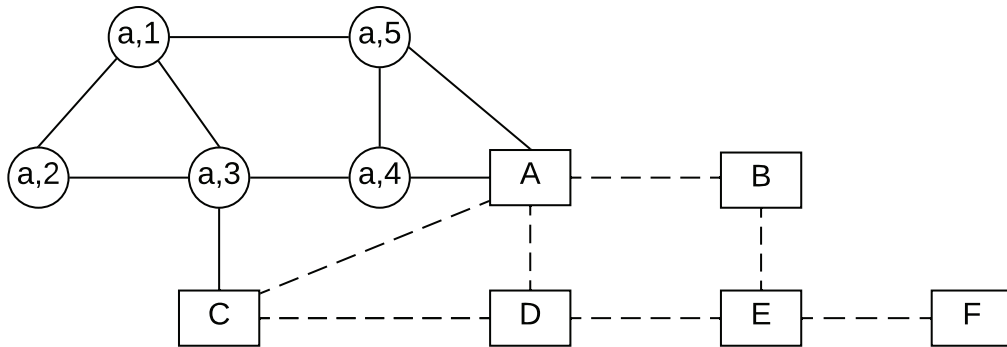


Figure 4.3: A high-level logistic map from agent  $a$ 's perspective

#### 4.5.2 Step 2: Each intermediate agent provides map and route information

After creating the high-level map, agent  $a$  asks the agents in the intermediate areas to provide route and map information. In **Figure 4.3**, the intermediate agents are  $b$ ,  $c$ ,  $d$  and  $e$ . To protect the topological privacy of local maps, each intermediate agent uses the Laplace mechanism to obfuscate its local map, i.e., modify the number of bases and routes. Moreover, to protect length privacy, each intermediate agent uses the exponential mechanism, along with an RL algorithm, to assign probability distributions over the routes on its obfuscated local map while removing the distance information. Finally, each

intermediate agent presents an obfuscated local map, with probability distributions over routes, to agent  $a$ . An example explaining this process is presented below.

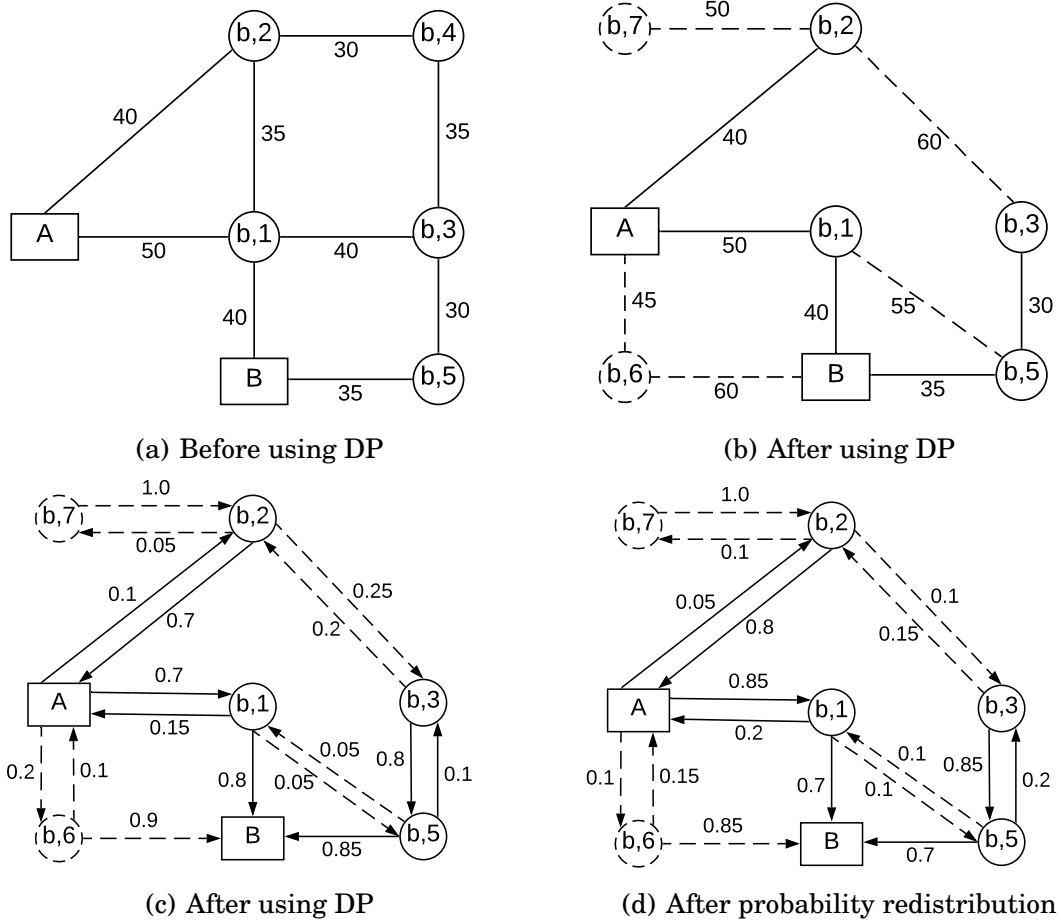


Figure 4.4: Obfuscation of agent  $b$ 's local map

In this example, **Figure 4.4(a)** is agent  $b$ 's local map with route length. **Figure 4.4(b)** is agent  $b$ 's obfuscated local map. Referring to the obfuscated local map, agent  $b$  calculates the shortest route between logistic centers  $A$  and  $B$ . Then, agent  $b$  marks the probability distributions over the routes, as shown in **Figure 4.4(c)**. Each probability on a route indicates the probability of that route being selected. To guarantee the route length privacy, agent  $b$  uses the exponential mechanism to redistribute these probabilities over the routes, as shown in **Figure 4.4(d)**. Agent  $b$  then sends **Figure 4.4(d)** to agent  $a$ . Finally, agent  $a$  receives a map where the topology has been obfuscated and the distance information has been replaced by probability distributions.

### 4.5.2.1 Using the Laplace mechanism to obfuscate topology

The Laplace mechanism is applied to the statistical information contained in a map. We utilize a  $1K$ -distribution [70] to obtain the statistical information. More specifically, the  $1K$ -distribution is used to calculate the node degree distribution of a given graph. To describe how the  $1K$ -distribution is utilized for this purpose, we employ the following example. In **Figure 4.4(a)**, the number of nodes with 1 degree is 0; the number of nodes with 2 degrees is 4, (i.e., nodes  $A, B, (b,4)$  and  $(b,5)$ ); the number of nodes with 3 degrees is 2, (i.e., nodes  $(b,2)$  and  $(b,3)$ ); and the number of nodes with 4 degrees is 1, (i.e., node  $(b,1)$ ). Thus, the  $1K$ -distribution, i.e., the node degree distribution, of **Figure 4.4(a)** is:  $P(1) = 0, P(2) = 4, P(3) = 2,$  and  $P(4) = 1$ .

---

**Algorithm 7** The Laplace mechanism-based obfuscation

---

- 1: /\*Take agent  $b$  as an example\*/
  - 2: **Input:** agent  $b$ 's map (**Figure 4.4(a)**);
  - 3: **Output:** agent  $b$ 's obfuscated map (**Figure 4.4(b)**);
  - 4: Use  $1K$ -distribution to obtain the statistical information of  $b$ 's map;
  - 5: **for**  $k = 1$  to  $d_{max}$  **do**
  - 6:      $\tilde{P}(k) \leftarrow P(k) + [Lap(\frac{\Delta S \cdot d_{max}}{\epsilon})]$ ;
  - 7: **end for**
  - 8: Rewire nodes to satisfy each  $\tilde{P}(k)$ ;
- 

The Laplace mechanism-based obfuscation is outlined in **Algorithm 7**. In Line 4, the statistical information of  $b$ 's map is obtained using the  $1K$ -distribution. In Lines 5-6, the Laplace noise is added to each  $P(k)$  in order to randomize the node degree distribution; accordingly the number of nodes now becomes  $\sum_{1 \leq k \leq d_{max}} \tilde{P}(k)$ . Here,  $d_{max}$  is the maximum node degree in a map, and  $d_{max} = 4$  in the example of **Figure 4.4(a)**. After adding Laplace noise, the node degree distribution could be as follows:  $\tilde{P}(1) = 1, \tilde{P}(2) = 2, \tilde{P}(3) = 5,$  and  $\tilde{P}(4) = 0$ . Next, in Line 7, nodes are rewired to satisfy each  $\tilde{P}(k)$ , where  $k \in \{1, \dots, d_{max}\}$ . The node rewiring is carried out using the graph model generator provided in [70]. After node rewiring is complete, fake routes may be introduced, such as route  $A \rightarrow (b,6)$  in **Figure 4.4(b)**. The length of a fake route is randomly generated based on the average length of the existing real routes.

The reason why the Laplace mechanism is used here is that our aim is to obfuscate the topology of each agent's local map by modifying the degree distribution. Since a degree distribution consists of a set of numbers, the Laplace mechanism is more appropriate here than the exponential mechanism which is mainly used for proportionally selecting an element from a set. It should also be noted at this point that the Laplace mechanism

may generate negative numbers. This, however, is not a problem in this chapter, as we need both positive and negative Laplace noise to ensure that our approach satisfies  $\epsilon$ -DP. Moreover, we adopt the Laplace mechanism to add noise to node degree distributions rather than directly adding noise to the number of nodes or edges. By adding noise to node degree distributions, our approach can not only guarantee the node and edge privacy of agents, but also guarantee the connection of an obfuscated graph. The connection of an obfuscated graph is a necessity for the completeness of our planning approach. The detailed theoretical analysis will be given in the next section.

The rationale behind **Algorithm 7** is as follows. According to the definitions of DP, a map is interpreted as a dataset  $D$ , while a node on a map is interpreted as a record in a dataset. As with the concept of neighboring datasets, two maps are deemed neighbors if they differ by only one node. Thus, using  $1K$ -distribution to obtain a map's statistical information can be thought of as querying some interesting information from a dataset,  $f(D)$ . If we compare Definition 2.6 to Line 6 in **Algorithm 7**, we can see that just as the Laplace mechanism can guarantee the privacy of a dataset, it can also guarantee the privacy of a map. More discussion about the preservation of privacy will be provided in the next section.

In **Algorithm 7**,  $\Delta S$  represents the sensitivity of the degree distribution in a map. The value of  $\Delta S$  is determined by the maximum change in degree distribution when a node is added into or removed from the map. For example, in **Figure 4.4(a)**, the degree scaling is from 1 to 4:  $P(1), P(2), P(3), P(4)$ . According to **Algorithm 7**, Line 6, when a node is added into or removed from the map, one of the four values,  $P(1), P(2), P(3), P(4)$ , will be incremented or decremented by 1. Thus, the maximum change of degree distribution is 1, i.e.,  $\Delta S = 1$  in **Algorithm 7**.

#### 4.5.2.2 Using reinforcement learning to compute probability distributions

In a local area, such as the one in **Figure 4.4(a)**, there is a set of local military bases and logistic centers, along with a set of routes connecting these bases and centers. As discussed in Section 4.4, in the Graph-STRIPS model,  $\mathcal{V}$  and  $\mathcal{E}$  can be used to represent the topology of a map. Accordingly, we use  $\mathcal{V}$  to represent the military bases and logistic centers, while  $\mathcal{E}$  is used to denote the set of routes connecting these bases and centers. Specifically, in **Figure 4.4(a)**,  $\mathcal{V}_b = \{(b, 1), (b, 2), (b, 3), (b, 4), (b, 5)\}$ , and  $\mathcal{E}_b = \{A \sim (b, 1), A \sim (b, 2), \dots, (b, 1) \sim B, (b, 5) \sim B\}$ . Moreover, different bases or centers will have different routes available to them. For example, in base  $(b, 1)$ , there are four available routes:  $\mathcal{E}_{(b,1)} = \{(b, 1) \sim A, (b, 1) \sim (b, 2), (b, 1) \sim (b, 3), (b, 1) \sim B\}$ . Furthermore, in center  $A$ , there

are two available routes:  $\mathcal{E}_A = \{A \sim (b, 1), A \sim (b, 2)\}$ .

---

**Algorithm 8** The reinforcement learning algorithm in map obfuscation

---

```

1: /*Take agent  $b$  as an example*/
2: Input: agent  $b$ 's obfuscated map (Figure 4.4(b));
3: Output: agent  $b$ 's obfuscated map with probability distributions (Figure 4.4(c));
4: Initialize probability distributions;
5: Initialize the Q-value of each route;
6: Initialize the current position:  $v \leftarrow A$ ;
7: while  $v \neq B$  do
8:   Agent  $b$  selects a route,  $e$ , based on the probability distribution  $\pi(v) = \langle \pi(v, e_1), \dots, \pi(v, e_n) \rangle$ , where  $e \in \mathcal{E}_v = \{e_1, \dots, e_n\}$ ;
9:    $r \leftarrow \mathcal{R}(v, e)$ ;
10:   $Q(v, e) \leftarrow (1 - \alpha)Q(v, e) + \alpha[r + \gamma \max_{e_i \in \mathcal{E}_v} Q(v', e_i)]$ ;
11:   $\bar{r} \leftarrow \sum_{e_i \in \mathcal{E}_v} \pi(v, e_i)Q(v, e_i)$ ;
12:  for each route  $e_i \in \mathcal{E}_v$  do
13:     $\pi(v, e_i) \leftarrow \pi(v, e_i) + \zeta(Q(v, e_i) - \bar{r})$ ;
14:  end for
15:   $\pi(v) \leftarrow \text{Normalise}(\pi(v))$ ;
16:   $v \leftarrow v'$ ;
17: end while
18: Agent  $b$  marks the learned probability distributions over the routes;
    
```

---

The RL algorithm is outlined in **Algorithm 8**. In Line 4, agent  $b$  proportionally initializes probability distributions over actions, where each action indicates the selection of a route. The initialization is based on the lengths of the routes. For example, in **Figure 4.4(b)**, the probability distribution over routes  $A \sim (b, 1)$  and  $A \sim (b, 6)$  can be initialized as  $\frac{4}{9}$  and  $\frac{5}{9}$ , respectively. In Line 5, agent  $b$  initializes the Q-value of each route; here, the Q-value is an indication of how good a route is. In this algorithm, the initial Q-value of a route is set based on the length of the route, such that a shorter route is allocated a higher Q-value. For example, in **Figure 4.4(b)**, the initial Q-value of route  $A \sim (b, 1)$  can be set to  $\frac{100}{50} = 2$ , while the initial Q-value of route  $A \sim (b, 6)$  can be set to  $\frac{100}{40} = 2.5$ . In Line 6, agent  $b$  sets the initial position to  $A$  and the destination to  $B$ . This setting is based on the fact that, as an intermediate agent, agent  $b$  will help agent  $a$  to transport the package from  $A$  to  $B$ .

Regarding the loop, in Line 8, agent  $b$  selects a route  $e$  based on the probability distribution over the available routes in base  $v$ . After taking route  $e$ , agent  $b$  receives a reward  $r$  (Line 9), which is inversely proportional to the route length. For example, in **Figure 4.4(b)**,  $r(A \sim (b, 1))$  and  $r(A \sim (b, 6))$  can be set to 4 and 5, respectively. The

reward  $r$  is used to update the Q-value of route  $e$  in base  $v$  (Line 10). This update is based on: 1) the current Q-value of  $e$  in base  $v$ ,  $Q(v, e)$ ; 2) the maximum Q-value of the routes in new base  $v'$ ,  $\max_{e_i \in \mathcal{E}_{v'}} Q(v', e_i)$ ; 3) the immediate reward  $r$ ; and 4) a learning rate  $\alpha$  and a discount rate  $\gamma$ . In the next step, the updated Q-value and the probability distribution are used to compute the average reward  $\bar{r}$  (Line 11), where  $\mathcal{E}_v$  is the set of available routes in base  $v$ . In Lines 12 and 13, the probability of selecting each route  $i \in \mathcal{E}_v$  is updated. This update is based on: 1) the current probability of each route being selected  $\pi(v, e_i)$ ; 2) the current Q-value of each route  $Q(v, e_i)$ ; 3) the average reward  $\bar{r}$ ; and 4) a learning rate  $\zeta$ . In Line 14, the updated probability distribution is normalized to be valid, meaning that for each  $i \in \mathcal{E}_v$ ,  $0 < \pi(v, e_i) < 1$  and  $\sum_{e_i \in \mathcal{E}_v} \pi(v, e_i) = 1$ . In Line 15, the new base,  $v'$ , is set as the current base. The above steps are iterated over until the goal state is reached. Finally, in Line 16, agent  $b$  marks each of the routes with the learned probability distributions.

#### 4.5.2.3 Using the exponential mechanism to redistribute probabilities

After using the RL algorithm to replace distance information with probability distributions, agents' local distance information can be hidden. Hiding distance information can reduce the risk of leaking this information but cannot guarantee the privacy preservation of this information. Therefore, we adopt the exponential mechanism to redistribute probabilities.

We use an example to explain how to use the exponential mechanism to redistribute probabilities. Suppose a node in a local map has two adjacent edges,  $x$  and  $y$ , and the probabilities of selecting  $x$  and  $y$  are 0.7 and 0.3, respectively. Based on the definition of exponential mechanism, the exponential mechanism selects and outputs an element  $r$  with probability proportional to  $\exp(\frac{\epsilon u_r}{2\Delta u})$ , where  $\epsilon$  is the privacy budget,  $u_r$  is the utility of selecting  $r$  and  $\Delta u$  is the sensitivity of utility. If we set the utility of selecting a route to be the probability of selecting that route, then we have:  $u_x = 0.7$  and  $u_y = 0.3$ , and in this setting,  $\Delta u = 1$ . Then, if we set  $\epsilon = 2$ , we have  $\exp(\frac{\epsilon u_x}{2\Delta u}) = 2.014$  and  $\exp(\frac{\epsilon u_y}{2\Delta u}) = 1.350$ . Finally, the probabilities of selecting  $x$  and  $y$  become  $\frac{2.014}{2.014+1.350} = 0.6$  and  $\frac{1.350}{2.014+1.350} = 0.4$ , respectively. The above process is performed on each node in the local map.

Another simple way to preserve the distance information privacy is to let each agent use the Dijkstra's algorithm [25] to compute the shortest route length between two logistic centers in its local area and add a Laplace noise to that length. However, other agents may still get an approximate idea about the route length. For example, after adding a Laplace noise, the route length changes from 100 to 105. Although other agents

cannot deduce the real length, they can still guess that the real length must be near 105. In some situations, e.g., the military logistic example, an approximate length is good enough for other agents. By contrast, if an agent uses RL and shares only probabilities, other agents cannot obtain even an approximate length. This idea is based on the spirit of federated learning by allowing agents to share only parameters [131]. In federated learning, to protect each client’s training data privacy, each client only sends the model parameters, trained based on her private data, to the server. The server, thus, has only clients’ model parameters without any clients’ private data.

### 4.5.3 Step 3: Creating a complete plan

After receiving obfuscated local maps from intermediate agents, agent  $a$  creates a logistic map by combining these obfuscated local maps, as shown in **Figure 4.5**. On each obfuscated local map, although both real and fake nodes and edges are involved, agent  $a$  is unable to determine whether a given node or edge is real. More detailed discussion on this matter will be presented in Section 4.6.

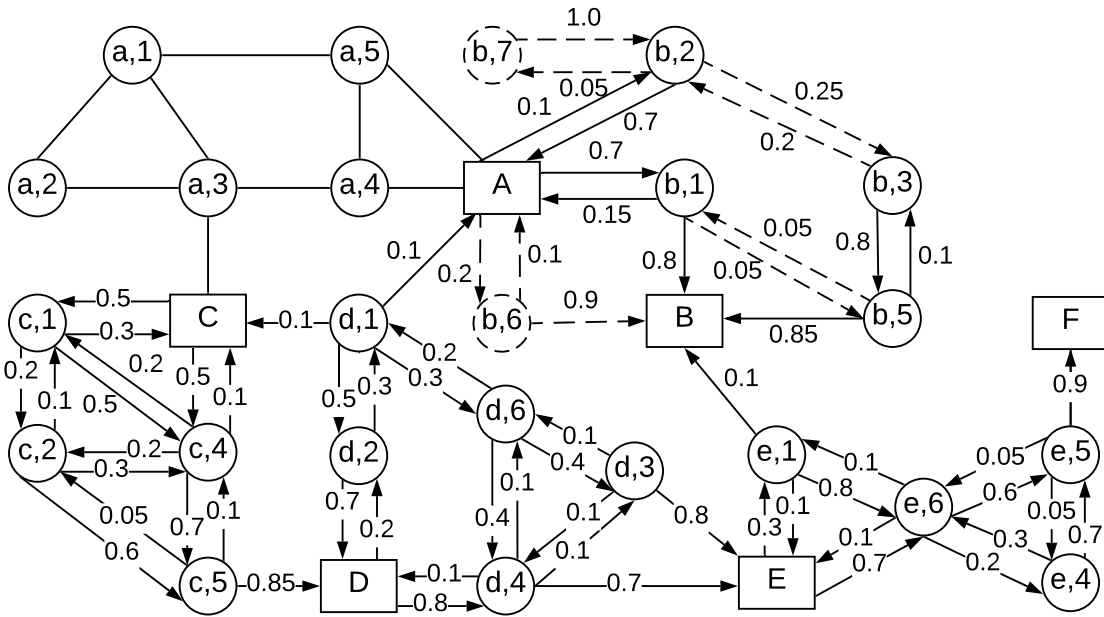


Figure 4.5: A logistic map created by obfuscated local maps

Next, agent  $a$  uses an RL algorithm to calculate the length of the route between each pair of connected logistic centers, e.g.,  $A \rightarrow B$ ,  $B \rightarrow E$  and so on. The RL algorithm is similar to **Algorithm 8**. Since agent  $a$  is only provided with probability distributions

about the other areas, agent  $a$  must generate the distance information itself based on the probability distributions. Agent  $a$  relates the probabilities to the distance based on the average route length in agent  $a$ 's local area. For example, in **Figure 4.5**, the probabilities of selecting routes  $A \sim (b, 1)$  and  $A \sim (b, 6)$  are 0.7 and 0.2, respectively. If the average route length in agent  $a$ 's local area is 45, agent  $a$  can simply set the distances from  $A$  to  $(b, 1)$  and  $A$  to  $(b, 6)$  to 20 and 70, respectively, whose average is 45. Here, we operate under the assumption that there are no significant differences between the average route length in each local area.

After agent  $a$  calculates the length of the shortest route between each pair of connected logistic centers (as shown in **Figure 4.6**), the shortest route from the origin to the destination can also be obtained. It is clear at this point that this calculation is not very accurate, as it is based on estimated length. However, the aim of this calculation is not to find the real shortest route, rather to select the intermediate agents which are located on the shortest route. In Fig 4.6, the agents on the shortest route are:  $b$ ,  $e$  and  $f$ .

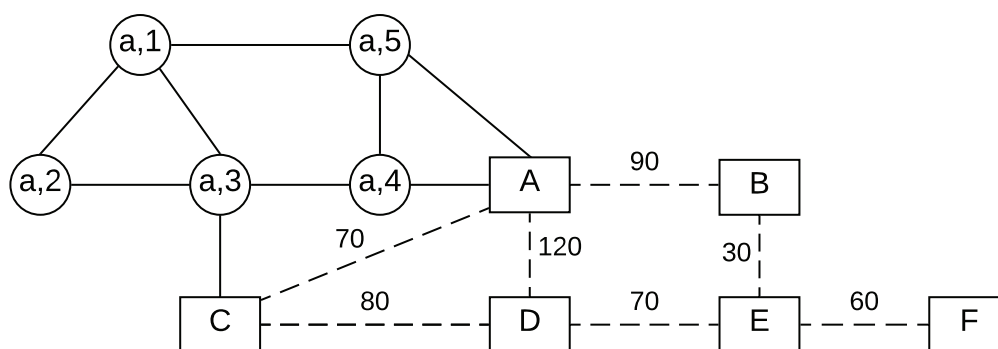


Figure 4.6: A high-level map featuring relative distances from agent  $a$ 's perspective

The final plan, thus, can be expressed as  $\Pi_a^\triangleright = \langle \mathcal{I} \rightarrow (a, 3) \rightarrow (a, 4) \rightarrow A \rightarrow B \rightarrow E \rightarrow F \rightarrow \mathcal{G} \rangle$ , where  $\mathcal{I} = \{package\_in\_ (a, 2)\}$  and  $\mathcal{G} = \{package\_in\_ (f, 4)\}$ . In this plan,  $\mathcal{I} \rightarrow (a, 3) \rightarrow (a, 4) \rightarrow A$  is the local plan formulated and carried out by agent  $a$ . At logistic center  $A$ , agent  $a$  gives its package to agent  $b$ , which makes a local plan to transport the package to logistic center  $B$ . At center  $B$ , agent  $e$  takes control of the package and devises a local plan to deliver the package to logistic center  $F$ . Finally, agent  $f$  picks up the package at center  $F$  and makes a local plan to transfer the package to  $(f, 4)$ .



#### 4.5.4 A simplification of the proposed approach

In some situations, if the distance information is not private, we can let logistic centers do the routing planning and consider the routing only between logistic centers. Each logistic center can directly communicate with the agent that is connected with the logistic center. As the distance information is not private, each logistic center is also aware of the local routing information within the agent. Compared with the proposed approach, this simplified approach can 1) significantly reduce the problem complexity; and 2) enable agents to obtain accurate distance information for further calculation; and 3) fully hide the topology information belonging to each agent from other agents.

A typical example is daily logistic, where the distance information between two public places do not need to be hidden. In daily logistic, packages are transported from their starting points to their destinations across multiple states or provinces. Here, the distance among states/provinces is not a privacy concern and can be considered as public information. The simplified version of our approach can be applied to this example. Each state/province is assumed to have a logistic center. To transport a package, the logistic center at the starting point utilizes the accurate distance information among states/provinces to make an optimal global plan. Then, each logistic center in the global planning path conducts the local routing planning.

### 4.6 Theoretical analysis

#### 4.6.1 Soundness analysis

**Theorem 4.1.** *The proposed approach is sound.*

**Proof.** We prove this theorem by considering one task, e.g., delivering one package in the logistics example. In Step 1 of our approach, we start from the initial agent which has a task to complete and initializes a plan, each queried agent sets up a link to the querying agent. Thus, all the queried agents are reachable. If a goal agent is identified whose private facts include the goal state, there must be at least one plan connecting the initial agent to the goal agent through some or all of the queried agents. ■

#### 4.6.2 Completeness analysis

**Lemma 4.1.** *Obfuscating local maps does not affect the completeness of the proposed approach.*

**Proof.** In Step 2 of our approach, each intermediate agent obfuscates its local map by adding and/or removing nodes and/or edges (see **Algorithm 7**). During the obfuscation process, Laplace noise is added to the node degree distribution of the local map:  $P(1), \dots, P(d_{max})$ . As  $P(0)$  is not counted, isolated nodes will not be created. Moreover, as the obfuscated map is undirected, it can be guaranteed that the obfuscated map will be connected. Hence, there must be at least one route between the two logistic centers on the local map. Since this property is common to the local maps of all intermediate areas, there must be at least one route from the initial area to the goal area via intermediate logistic centers. Thus, the completeness is not affected. ■

**Theorem 4.2.** *The proposed approach is complete.*

**Proof.** Step 1 of our approach guarantees that a goal agent can be found. According to **Theorem 4.1**, there must be at least one plan connecting the initial agent to the goal agent. We now need only to prove that our approach is capable of finding at least one of these plans. According to Lemma 4.1, there is at least one route from the initial area to the goal area. One of these routes can be treated as a high-level plan, which can be identified using **Algorithm 8**. Based on the high-level plan, each intermediate agent creates a local plan (Step 3). Given that each agent is honest<sup>1</sup>, each local plan is valid, which ensures that the two logistic centers in the local area will be connected. Therefore, a high-level plan and a set of local plans constitute a complete plan. ■

### 4.6.3 Privacy-preserving analysis

**Theorem 4.3.** *The proposed planning approach satisfies  $\epsilon$ -DP.*

**Proof.** To analyze the privacy guarantee, we apply two composite properties of the privacy budget: the sequential and the parallel compositions [75]. The sequential composition determines the privacy budget  $\epsilon$  of each step when a series of private analysis are performed sequentially on a dataset. The parallel composition corresponds to the case in which each private step is applied to disjoint subsets of a dataset. The ultimate privacy guarantee depends on the step which has the maximal  $\epsilon$ .

In the proposed approach, the Laplace mechanism and the exponential mechanism consumes the privacy budget. In the Laplace mechanism in **Algorithm 7**, the Laplace noise sampled from  $Lap(\frac{\Delta S \cdot d_{max}}{\epsilon})$  is added in  $d_{max}$  steps. At each step, the Laplace

---

<sup>1</sup>It is a common assumption in privacy-preserving multi-agent planning that agents are honest but curious about others' private information [118].

mechanism consumes the  $\frac{\epsilon}{d_{max}}$  privacy budget; thus for each step, **Algorithm 7** satisfies  $\frac{\epsilon}{d_{max}}$ -DP. By using the sequential composition property, we can conclude that at a total of  $d_{max}$  steps, the Laplace mechanism consumes the  $d_{max} \cdot \frac{\epsilon}{d_{max}} = \epsilon$  privacy budget, meaning that **Algorithm 7** satisfies  $\epsilon$ -DP. By comparing Definition 2.2 with Definition 4.5, since the Laplace mechanism can guarantee the data record privacy of a dataset, it can also guarantee the node-privacy of a graph.

The exponential mechanism is used to redistribute probabilities on each agent's local graph. For a given node in a local graph, suppose the node has  $k$  adjacent edges. Then, the exponential mechanism will be used  $k$  times. If we set privacy budget for this node to be  $\frac{\epsilon}{k}$ , based on the sequential composition property, the privacy consumption of this node is  $\epsilon$ . Thus, the probability redistribution on the adjacent edges of this node satisfies  $\epsilon$ -DP. When this method is used on every node, based on the parallel composition property, the probability redistribution on this local graph satisfies  $\epsilon$ -DP.

Since the Laplace mechanism and the exponential mechanism are used by each agent, each agent is guaranteed  $\epsilon$ -DP. Although an environment may contain multiple agents, each agent maintains a local area, and these local areas are disjoint with each other. Since each agent is guaranteed  $\epsilon$ -DP, according to the parallel composition property, the proposed approach satisfies  $\epsilon$ -DP. ■

**Remark 1:** In **Algorithm 7**, Laplace noise is used to randomize the node degree distribution. This implies that both the number of nodes and the number of edges in a local map will be perturbed. Since the topology of a map consists of nodes and edges, perturbing the numbers of nodes and edges incurs perturbation of the topology. Accordingly, as **Algorithm 7** satisfies DP, the perturbation of the topology of a map also satisfies DP.

**Corollary 4.1.** *No agent is able to conclude anything about the existence of any subset of  $\lceil \frac{\Delta S \cdot d_{max}}{\epsilon} \rceil$  nodes in another agent's map.*

**Proof.** In **Algorithm 7**, the Laplace noise is sampled from  $Lap(\frac{\Delta S \cdot d_{max}}{\epsilon})$ , meaning that the expected amount of noise is  $\frac{\Delta S \cdot d_{max}}{\epsilon}$ . As this noise is used to change the number of nodes in a map (recall Lines 5-6 in **Algorithm 7**), the expected number of nodes that will be changed is  $\lceil \frac{\Delta S \cdot d_{max}}{\epsilon} \rceil$ . Therefore, any subset of  $\lceil \frac{\Delta S \cdot d_{max}}{\epsilon} \rceil$  nodes could be fake nodes. According to Definition 4.5 and **Theorem 4.3**, since **Algorithm 7** can guarantee the node-privacy of a graph, an agent will be unable to distinguish real from fake statistical information between two neighboring graphs, e.g., the number of real nodes. This means that an agent cannot determine whether or not a node is fake. Hence, the existence of

any subset of  $\lceil \frac{\Delta S \cdot d_{max}}{\epsilon} \rceil$  nodes in an agent's map cannot be concluded by any other agents.

■

**Remark 2:** From Corollary 4.1, in the Laplace mechanism in **Algorithm 7**, the value of  $\epsilon$  controls the granularity of privacy, given that the values of  $\Delta S$  and  $d_{max}$  have been fixed. A smaller  $\epsilon$  implies a stronger privacy guarantee. However, a smaller  $\epsilon$  also introduces a larger amount of noise. The increase of the amount of noise reduces the usability of a map. Thus, the value of  $\epsilon$  should be carefully set.

**Remark 3:** Similar to the Laplace mechanism, in the exponential mechanism, the value of  $\epsilon$  has a huge impact on probability redistribution results. Given that a node has  $k$  adjacent edges and the probabilities of selecting the  $k$  edges are  $u_1, \dots, u_k$ , if we set  $\epsilon = 0$ , the probability of selecting each edge will equally become  $\frac{1}{k}$ ; if we set  $\epsilon \rightarrow +\infty$ , probability  $u_m$  becomes 1 and others become 0, where  $u_m = \max\{u_1, \dots, u_k\}$ . In addition to the two extreme situations, there is a median situation which is that the redistributed probabilities are identical to the original probabilities:  $u'_1 = u_1, \dots, u'_k = u_k$ . Based on the computation method described in Section 4.5.2, each probability  $u'_i$ ,  $1 \leq i \leq k$ , is computed as:

$$(4.2) \quad u'_i = \frac{\exp(\frac{\epsilon u_i}{2\Delta u})}{\sum_{1 \leq j \leq k} \exp(\frac{\epsilon u_j}{2\Delta u})}.$$

Let each  $u'_i = u_i$ , we have  $k$  equations.

$$\left\{ \begin{array}{l} \frac{\exp(\frac{\epsilon u_1}{2\Delta u})}{\sum_{1 \leq j \leq k} \exp(\frac{\epsilon u_j}{2\Delta u})} = u_1, \\ \dots, \\ \frac{\exp(\frac{\epsilon u_k}{2\Delta u})}{\sum_{1 \leq j \leq k} \exp(\frac{\epsilon u_j}{2\Delta u})} = u_k. \end{array} \right.$$

In our problem,  $\Delta u = 1$ . By solving the  $k$  equations, we have that

$$\epsilon_i = \frac{2(k \cdot \ln(u_i) - \sum_{1 \leq j \leq k} \ln(u_j))}{k \cdot u_i - \sum_{1 \leq j \leq k} u_j},$$

where  $1 \leq i \leq k$ . Thus, in applications, on one hand, these values of  $\epsilon$  should be avoided, as they will make the redistributed probabilities identical to the original probabilities, which cannot offer any privacy preservation. On the other hand, the values of  $\epsilon$  should be set close to these values to guarantee the usability of the redistributed probabilities.

**Remark 3:** The privacy guarantee secured by adding noise to the number of nodes is stronger than that arising from adding noise to the number of edges. This is because

adding noise to the number of nodes results in more distortion than does adding noise to the number of edges. For example, adding a node to a map results in the addition of up to  $m$  edges to the map, where  $m$  is the number of nodes in the map. By contrast, adding an edge to a map results in the addition to the map of one node at most. More distortion leads to a stronger privacy guarantee.

**Theorem 4.4.** *The proposed planning approach can strongly preserve agents' privacy.*

**Proof.** As defined in Section 4.4.1, an agent's private information includes 1) the number of nodes in an agent's local area, 2) the number of edges in the local area, 3) the length of these edges, 4) the positions of any items in the local area and 5) the movements of any items in the local area. To prove this theorem, we only need to prove that the private information possessed by an agent cannot be inferred by another agent. First, according to **Theorem 4.3** and Corollary 4.1, the proposed planning approach satisfies  $\epsilon$ -DP and guarantees the privacy of any subset of  $\lceil \frac{\Delta S \cdot d_{max}}{\epsilon} \rceil$  nodes in an agent's local area. By properly setting the value of  $\epsilon$ , the privacy of all nodes and edges in an agent's local area can be preserved. Therefore, the privacy of the number of nodes and edges of an agent's local area will also be preserved.

Second, our approach dictates that the length information in a local area is replaced by probability distributions (recall **Figure 4.4**). Also, these probabilities are redistributed using the exponential mechanism. Thus, the length information is strictly hidden. Therefore, an agent cannot infer the real length of any individual edge in another agent's local area. Third, since the privacy of any node or edge in an agent's local area has been preserved, the positions and movements of items have also been preserved. Based on the definition of strong privacy (Definition 4.2), the proposed approach can strongly preserve agents' privacy. ■

#### 4.6.4 Communication analysis

Let us suppose that there are  $m$  logistic centers. Each logistic center,  $i$ , has a capacity,  $lc_i$ , which is the maximum number of agents that can share the logistic center. Accordingly, we derive the following theorem:

**Theorem 4.5.** *In Step 1, the upper bound of the number of communication messages used to find a goal agent is  $\sum_{1 \leq i \leq m} lc_i$ .*

**Proof.** In our approach, each agent is only aware of the existence of its own neighbors. This means that 1) each agent does not know how many neighbors any other agent has,

and 2) each agent is not aware of how far away the goal agent is. As the information regarding logistic centers is public, all agents know the capacity of each logistic center. Thus, to guarantee that the query message is able to reach the goal agent, an agent must assume that 1) each logistic center is using up its capacity, and 2) the goal agent is located in the most distant area. In this situation, the number of generated communication messages is  $\sum_{1 \leq i \leq m} lc_i$ . ■

**Remark 4:** **Theorem 4.5** describes the communication overhead in the worst case. However, as time progresses, this communication overhead can be significantly reduced. This is because an agent memorizes the plans that it has previously created, meaning that an agent memorizes the routes to goal agents. Thus, in the future, an agent can simply exploit a route previously determined to reach a goal agent without the need for communication. Even if an agent decides to explore a new route, the communication overhead can be limited by setting the maximum number of query messages during the finding process. The maximum number of query messages is set to be identical to the number of messages used to find the same goal agent last time. Formally, we have the following corollary:

**Corollary 4.2.** *As time progresses, the communication overhead of each agent monotonically decreases.*

**Proof.** Every time an agent explores a new route to a goal agent, the maximum number of query messages is set to be equal to the number of messages used to find the same goal agent last time. As each agent memorizes only the shortest routes to goal agents, only routes that are shorter than these memorized routes will be taken by each agent. This means that the number of request messages currently being used must be fewer than or equal to the number used previously. Thus, the communication overhead of each agent monotonically decreases. ■

In our approach, the setting of the communication budget  $C$  can be controlled by the privacy budget  $\epsilon$ . In a multi-agent system, each agent  $k$  sets  $\epsilon/C$  as their privacy budget and ceases to communicate when  $\epsilon$  is used up. When  $C > \sum_{1 \leq i \leq m} lc_i$ , the system can guarantee that all communication steps will be completed. However, a large amount of noise will be added to the system under these circumstances. When  $C < \sum_{1 \leq i \leq m} lc_i$ , the system is likely to stop before finishing the communication steps. However, the noise added to the system will be limited. When  $C = \sum_{1 \leq i \leq m} lc_i$ , the system will stop when all communication steps have been completed. Therefore, by adjusting the privacy budget  $\epsilon$

and the communication budget  $C$ , the communication overhead of a multi-agent system can be controlled.

## **4.7 Application of our approach to other domains**

This section illustrates how our approach can be applied to three other domains: networks, air travel, and rovers.

### **4.7.1 Packet routing in networks**

In a network, nodes often transmit packets between each other. These nodes may belong to different areas, which are connected by routers or access points. In this domain, a router or access point can be thought of as similar to an agent, which manages a corresponding area. In a given area, the information possessed by each node, e.g., its load and performance, is private to the agent. Moreover, the number of nodes in an area and their communication links are also private to the agent. Thus, the agents expect that their privacy will be preserved.

As each node has only a limited range of communication, when a node transmits a packet to another node, the packet may be relayed multiple times by intermediate nodes before reaching its destination. Since it is highly desirable that nodes receive packets in a timely manner, the transmission must be efficient so that huge delays can be avoided. The proposed approach can be applied to create efficient plans for packet routing.

### **4.7.2 Airplane transport**

The airplane transport problem consists of a set of planes and airports. Moreover, the travel map is partitioned into a set of areas. In the real world, each area can be thought of as a country. Therefore, the planes and airports located in a given area are private to the area air traffic controller. Clearly, each area controller wants to preserve information regarding the status and number of planes and airports in their area as private information.

The airports located on the boundary of two areas are public. The goal is to transport passengers between airports. In this problem, each area controller can be thought of as an agent. When a plane travels from one airport to another, as the plane has only limited fuel, passengers may be transferred multiple times on their way to their destination. Moreover, both area controllers and passengers would clearly prefer the plane to reach

its destination as quickly as possible. Thus, an efficient privacy-preserving planning approach is required. The proposed approach can be applied to create efficient plans for passenger transport.

### **4.7.3 Rover exploration**

This domain models Mars exploration rovers. Each rover can be thought of as an agent. The goal of these rovers is to collect samples. Each rover has its own private sets of targets and reachable locations. These targets and reachable locations can be thought of as private facts in our planning model, the privacy of which must be preserved.

Each rover collects samples in its reachable locations. When a rover needs to transmit the samples it has collected to another rover, these samples may have to be transmitted by intermediate rovers in the interim, as the number of locations reachable by each rover is limited. Since samples may decay as time progresses, it is desirable for the rovers to transmit the samples to the destination as quickly as possible. Hence, an efficient privacy-preserving planning approach is required. The proposed approach can be applied to create efficient plans for sample transmission.

In summary, our approach can be applied to all of the planning problems, in which each party has private information and local plans can be created by each party using reinforcement learning techniques. Moreover, reinforcement learning has a broad range of applications, including task scheduling in cloud computing [91], traffic light control [4], and robot coordination [58]. Since most of these applications may also have privacy requirements, our method has the potential to be applied to these real-world scheduling and coordination problems as well.

## **4.8 Experiments**

### **4.8.1 Experimental setup**

The experiments in the present research are conducted based on two scenarios: logistics and packet routing, which are typical logistic-like problems. In the logistics scenario, as described in Section 4.3, each military base has a set of packages to transport to other military bases. These military bases may be located in different areas and managed by different military units. The information pertaining to each military base is private to the managing military unit.



The packet routing scenario is similar to the logistics scenario, in that each node in an ad hoc network houses a set of packets to be sent to other nodes. Nodes may belong to different groups and are served by different access points. The information of each node is private to the serving access point. The key difference between these two scenarios is that in the packet routing scenario, new nodes may dynamically join the network and existing nodes may leave the network at any time, while this is not the case for the logistics scenario. These experiments have also been conducted on the air travel and rover scenarios. As the results present a similar trend to logistics, they are not discussed here.

Three evaluation metrics are used in the two scenarios: 1) average route length: the average length of the routes from initial states to goal states; 2) average communication overhead: the average number of communication messages used to make a plan; 3) success rate: the ratio of the number of the successfully transmitted packages/packets to the total number of packages/packets.

In both scenarios, the map shape or network topology is similar to that in **Figure 4.1**. The size of the maps/networks varies from 10 logistic centers/access points to 50 logistic centers/access points; correspondingly the number of military bases/network nodes varies from 50 to 250 <sup>2</sup>.

The probability of a package/packet being generated on each military base/node is set to 0.2. The communication budget of each agent varies from  $C = 40$  to  $C = 80$  depending on variations in the map/network size. The privacy budget of each agent is set to  $\epsilon = 0.5$ . Moreover, in the packet routing scenario, during the route finding process, there is a probability of 0.1 that an existing node will leave the network and a probability of 0.1 that a new node will join the network. The parameter values in the proposed algorithms are chosen experimentally, and set to  $\alpha = 0.1$ ,  $\gamma = 0.9$  and  $\zeta = 0.95$ .

The proposed planning approach, denoted as *DP-based*, is evaluated in comparison with three closely related approaches. The first approach, denoted as *No-privacy*, is also developed by us. The major features of *No-privacy* are the same as *DP-based*, but the privacy-preserving mechanism has been removed. Although *No-privacy* is not applicable to privacy-preserving planning, it can be used to evaluate how the privacy-preserving mechanism impacts the performance of our *DP-based* approach. The second approach is based on best-first forward search, denoted as *Best-first*, and has been used in [12, 82, 109]. In the *Best-first* approach, when an agent transmits a package/packet to

---

<sup>2</sup>The topologies of maps/networks are created by simulation, as most real-world graph datasets [63] do not contain distance information and thus cannot be used in our experiments. We leave the experiments with real-world datasets as one of our future studies.

a logistic center/access point, the agent broadcasts this state to all the other agents. The nearest agent takes the package/package based on this state and transmits it to the next logistic center/access point. This process continues until the goal agent is reached. The third approach is GPPP (greedy privacy-preserving planner), denoted as *Greedy*, which was developed in [73]. The *Greedy* approach consists of two phases: global planning and local planning. In the global planning phase, all agents collaboratively devise a global plan using a best-first search method. Next, in the local planning phase, each agent creates a local plan by executing a single-agent planning procedure.

## 4.8.2 Experimental results

### 4.8.2.1 The logistics scenario

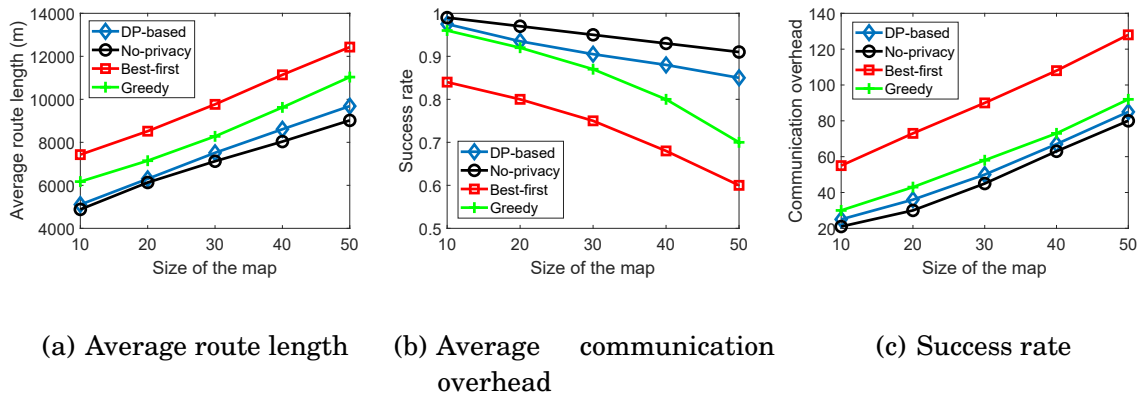


Figure 4.7: Performance of the four approaches on the logistics scenario with variation of the map size

**Figure 4.7** demonstrates the performance of the four approaches on the logistics scenario with variation of the map size. As the map size grows larger, for all four approaches, the average route length and the average communication overhead progressively increase, while the success rate gradually decreases. As the map size increases, the distance between an original agent and a destination agent may be enlarged accordingly. Therefore, the average route length increases. Moreover, when this occurs, the number of intermediate agents also increases. Thus, the average communication overhead rises as well. Due to this increase in the average communication overhead, the communication budget of some agents may be used up before a plan is made. Hence, the success rate reduces.

The proposed *DP-based* approach achieves much better performance than the *Best-first* and *Greedy* approaches. The reinforcement learning algorithm in the *DP-based* approach can find shorter routes than the other two approaches. Moreover, in the *DP-based* approach, agents are allowed to communicate only with neighbors, and a privacy budget is adopted to control communication overhead. Thus, the *DP-based* approach uses less communication overhead than the other two approaches. In addition, the *DP-based* approach successfully makes more plans than the other two approaches before the communication budget is used up. Overall, the performance of *No-privacy* approach is slightly better than the *DP-based* approach. As privacy is not taken into account in the *No-privacy* approach, the information shared between agents is accurate, and agents can make accurate plans based on this accurate information. However, the private information of each agent is entirely disclosed to other agents under this approach, a situation that should be avoided in real-world applications. More specifically, the average route length in the *DP-based* approach is only about 2% longer than for the *No-privacy* approach. This is because in the *DP-based* approach, a plan is made up of a set of local plans created by the initial agent and the intermediate agents. Each of these local plans is created by an individual agent with reference to its private but accurate information. Since most of the information used to create a plan is accurate, the introduction of our privacy-preserving mechanism does not substantially impact the average route length.

The *Best-first* approach achieves the worst performance out of the four approaches. In the *Best-first* approach, a package is transmitted to the nearest agent. However, in large and complex maps, the nearest agent may not always be the best choice. Moreover, always choosing the nearest agent may result in a transmission loop; if this situation arises, packages will never reach their destinations. In comparison, the performance of *Greedy* approach is better than the *Best-first* approach, as the *Greedy* approach features a global planning phase that involves selecting the appropriate logistic centers to create a high-level route, which conserves communication overhead.

**Figure 4.8** demonstrates the performance of the *DP-based* approach on the logistics scenario with variation of the privacy budget  $\epsilon$  value from 0.2 to 0.8. The number of logistic centers is fixed at 10. It can be seen that with the increase of the privacy budget  $\epsilon$  value, the performance of the *DP-based* approach improves, namely it achieves a shorter average route length (**Figure 4.8(a)**), lower average communication overhead (**Figure 4.8(b)**), and higher success rate (**Figure 4.8(c)**). According to the Laplace mechanism, when the  $\epsilon$  value is small, the noise, added to the map, is large. A large noise value will significantly affect the agents planning. For example, agent  $a$  has two neighbors  $b$  and

CHAPTER 4. DIFFERENTIALLY PRIVATE MULTI-AGENT PLANNING FOR LOGISTIC-LIKE PROBLEMS

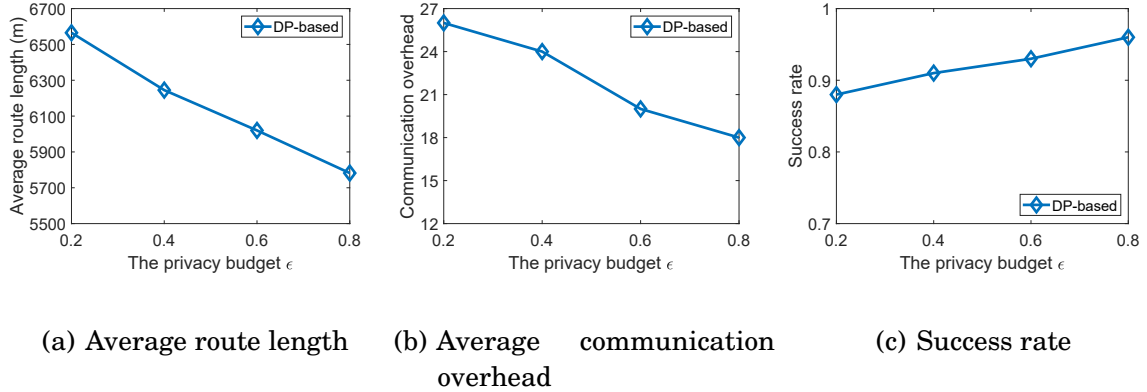


Figure 4.8: Performance of the *DP-based* approach on the logistics scenario with variation of the privacy budget value

c. Now, suppose that 1) agent  $a$  wants to send a package to  $d$ , and 2) delegating the package to  $b$  is a better choice than  $c$ . However, when agents  $b$  and  $c$  obfuscate their maps, due to the large noise, the obfuscation results may make  $c$  appear to be a better choice than  $b$ . Thus, agent  $a$  may make a sub-optimal plan. This situation is alleviated when the  $\epsilon$  value increases.

4.8.2.2 The packet routing scenario

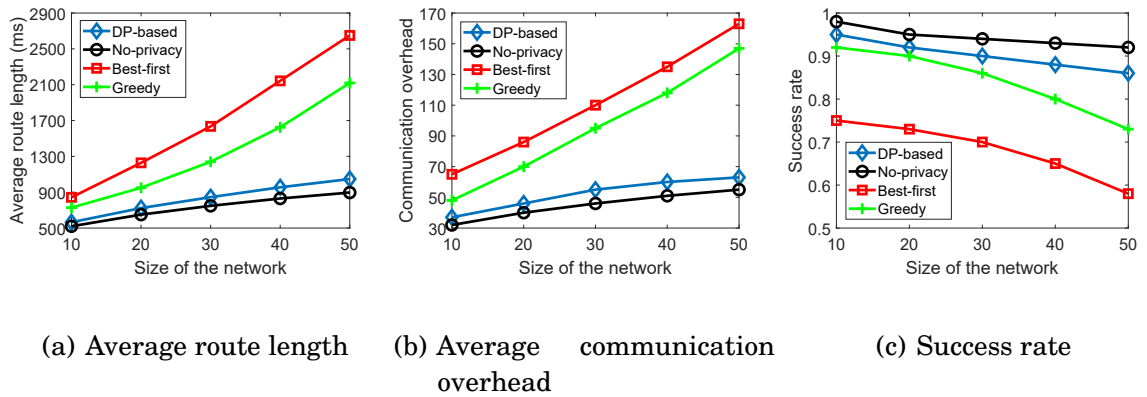
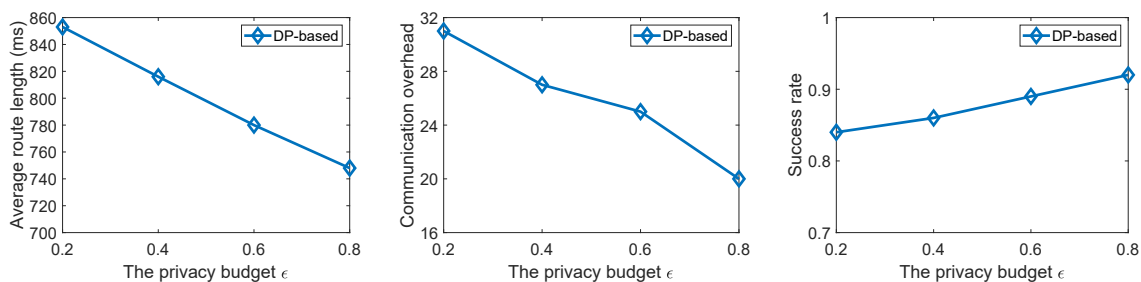


Figure 4.9: Performance of the four approaches on the packet routing scenario with variation of the network size

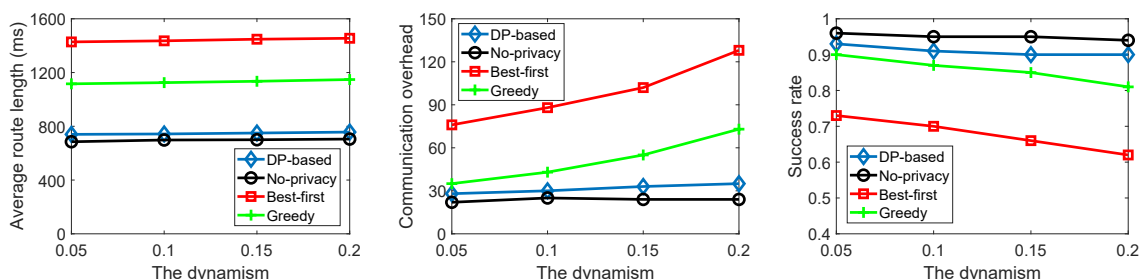
**Figure 4.9** illustrates the performance of the four approaches on the packet routing scenario with variation of the network size, while **Figure 4.10** depicts the performance of the *DP-based* approach on the packet routing scenario with variation of the privacy budget  $\epsilon$ . After comparing **Figure 4.7** to **Figure 4.9** and **Figure 4.8** to **Figure 4.10**, it



(a) Average route length (b) Average communication overhead (c) Success rate

Figure 4.10: Performance of the *DP-based* approach on the packet routing scenario with variation of the privacy budget

can be concluded that these approaches exhibit similar trends in terms of their results on the two scenarios, but that the performance of these approaches is worse on the packet routing scenario than on the logistic scenario. This is mainly due to the dynamism of the packet routing scenario. When a node leaves the network, the routes involving that node are broken. Thus, agents have to re-find routes. This incurs extra communication overhead and reduces success rates to some extent.



(a) Average route length (b) Average communication overhead (c) Success rate

Figure 4.11: Performance of the three approaches on the packet routing scenario with variation of the dynamism

**Figure 4.11** illustrates the performance of the four approaches on the packet routing scenario with variation of the dynamism, such that the probability of a node leaving or joining the network varies from 0.05 to 0.2 and the network size is fixed at 10 access points. From **Figure 4.11**, it can be seen that an increase in the dynamism negatively affects the *Best-first* and *Greedy* approaches in terms of their average communication

overhead and success rates, but does not significantly impact the *DP-based* and *No-privacy* approaches.

As the dynamism increases, the frequency with which nodes leave or join the network also increases. Thus, the number of affected routes increases as well. In the *Best-first* and *Greedy* approaches, when a route is broken, a new finding process is launched. This may not significantly affect the average route length (**Figure 4.11(a)**), as route length depends on the positions of nodes rather than the number of nodes. However, launching a new finding process results in additional communication overhead, and may thus reduce success rates due to depletion of the communication budget. By contrast, the *DP-based* and *No-privacy* approaches do not require a new finding process when a route is broken. In the *DP-based* approach, routes are found by using reinforcement learning on obfuscated local network topologies. These obfuscated local network topologies are obtained using DP. DP can guarantee that a node being brought in or out of a local network will have minimum effect on the statistical information. Therefore, when a node leaves or joins a local network, the serving access point does not need to re-obfuscate the new network or to communicate with the original access point about the change in the network. Hence, the communication budget can be conserved, and the success rate is preserved.

### 4.8.3 Summary

According to the experimental results, the proposed *DP-based* approach achieves better results than the *Best-first* and *Greedy* approaches in all experimental situations considered here. The average length of routes found by the *DP-based* approach is about 25% and 15% shorter, respectively, than those found using the *Best-first* and *Greedy* approaches. The *DP-based* approach also uses about 20% and 10% less communication overhead than the *Best-first* and *Greedy* approaches, respectively. Moreover, the *DP-based* approach achieves about 10% and 5% higher success rates than the *Best-first* and *Greedy* approaches, respectively.

Regarding performance, the *DP-based* approach is slightly worse than the *No-privacy* approach by a factor of about 3% in terms of average route length, 2% in communication overhead and 2% in success rate. The *DP-based* approach, however, strongly protects the privacy of agents, which is entirely disregarded in the *No-privacy* approach. Therefore, based on the experimental results, the efficiency of our *DP-based* approach can be proven.

## 4.9 Summary and Future Work

This chapter proposes a novel strong privacy-preserving planning approach for logistic-like problems. In this approach, an agent creates a complete plan by using obfuscated private information from each intermediate agent, where this obfuscation is achieved by adopting the DP technique. Due to the advantages of DP, following obfuscation, an agent's private information cannot be deduced by other agents regardless of their reasoning power. This approach is the first in existence to achieve strong privacy, completeness and efficiency simultaneously by taking advantage of DP. Moreover, this approach is communication-efficient. Compared to the benchmark approaches, our approach achieves better performance in various aspects.

In the future, we intend to extend our approach by introducing malicious agents. Existing approaches commonly assume that agents are honest but curious. Introducing malicious agents, which provide false information to others, may be a challenging and interesting addition to the field of multi-agent planning. Also, as described in the experimental part, we will continue to search usable real-world datasets and evaluate our approach with them.





## PRIVACY PRESERVATION IN DEEP REINFORCEMENT LEARNING: A TRAINING PERSPECTIVE

Reinforcement learning (RL) is a principled AI framework for autonomously experience-driven learning. DRL promotes a higher-level understanding of the visual world by incorporating deep learning models. Concerns related to privacy preservation in the context of RL are emerging due to an increasing number of applications involving a huge amount of private information. Some recent studies have demonstrated that DRL can leak private information and be vulnerable to attacks that attempt to infer the training environment from a training agent’s behaviors without accessing the environment. To address these privacy concerns, we propose a differentially private DRL approach to obfuscate the agent’s observations from each visited state, thereby preventing the inference of the agent’s training environment from its optimized policy and defending against privacy leakage attacks. We provide detailed theoretical analysis and design comprehensive experiments in a grid world environment to maximally reproduce the privacy leakage attack. Both theoretical analysis and experimental results demonstrate that our method can effectively defend against privacy leakage attacks as well as guarantee the model utility of the RL agent.

## 5.1 Introduction

Since deep reinforcement learning (DRL) is achieving great success in an increasing number of application fields that may involve huge amounts of private information, the security of policies and privacy preservation in DRL models have given rise to widespread concerns. For example, it has been demonstrated that DRL can leak private information about the training environment. An illustrative example presented by Pan et al. [86] concerns an agent that aims to navigate the shortest path between a starting point and a destination in a simple grid world with obstacles. As the example in **Figure 5.1** shows, a well-trained agent will still follow the same trajectory even once all obstacles in the environment have been removed after training. This example indicates that it is possible to infer environmental information from a well-trained DRL policy. A DRL agent tends to memorize the training environment instead of performing visual navigation. An adversary can thus recover the position of obstacles in the grid environment from the agent’s memory without needing to access the training environment directly. Similarly, Ye et al. [136] considered another concern related to privacy leakage in a multi-agent RL environment, specifically that an RL agent can leak information in the training environment of each party when multiple parties collaboratively train the RL agent. Therefore, it is possible for both conventional RL and DRL policies to leak sensitive information about a private training environment.

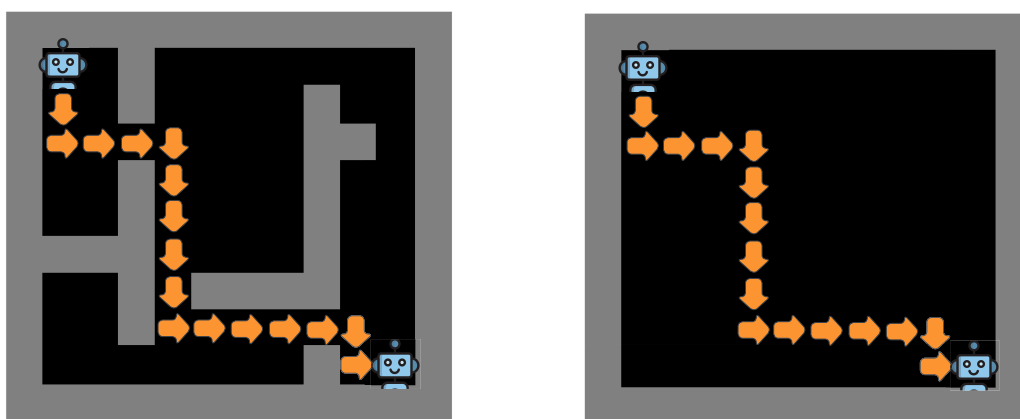


Figure 5.1: An example of trajectory disclosure from a well-trained DRL agent

We contend that the root cause of privacy leakage in DRL is the agent’s observations of the training environment. When an agent starts learning its task in a new DRL environment, the agent takes actions based on what it observes from the environment at every state. The agent obtains rewards from selected actions and links these rewards to

observations of its environment. Through repeatedly observing and learning about the environment, the agent takes actions with increasing confidence and tends to remember the sequence of actions it takes alongside its memory of the environment. When the agent is brought into a new open environment, the agent repeats the sequence of actions from memory, meaning that private information concerning its trajectory and the original environment can be inferred. This raises an interesting question: can the agent train its policy to complete the task successfully, but with inaccurate observations to prevent the environmental information from being leaked?

DP has the potential to provide a privacy-preserving solution. DP offers a strong privacy guarantee through the addition of perturbation to a pair of datasets, meaning that the results of queries of two neighboring datasets are likely to be statistically similar [103]. However, it is difficult to directly apply DP to the DRL in terms of the agent policy for the following reasons. First, it is a challenge for DP to balance the privacy guarantee and model utility. More specifically, DP has a stronger privacy guarantee with larger randomization, but this large randomization may have a huge negative impact on model utility. Conversely, small randomization can only provide a limited privacy guarantee. Thus, it is important to find a way to balance both privacy and utility simultaneously. Second, estimating how suitable randomization might be arranged in the learning process is no easy task. In other words, we have to estimate how ‘good’ the performance is in terms of DP.

To solve the above issues, we propose the *Differentially Private Deep Reinforcement Learning (DP-DRL)* method to protect a DRL agent’s training environment information against privacy leakage attacks. We apply the exponential mechanism of DP to protect an agent’s observations from each visited state by obfuscating one observation element each time based on a probability distribution. While considering the vector of a DRL agent’s observations as a dataset, if we apply DP to obfuscate one observation of the vector, the original observations and obfuscated observations in the agents training process will be statistically similar. Moreover, we dynamically adjust the privacy budget of DP to maximally guarantee model utility based on the impact on model utility from obfuscation, as well as providing the privacy guarantee. The proposed method can guarantee that the DRL agent can successfully train a well-performed policy in a privacy-preserving environment without information being leaked due to a privacy leakage attack. The main contributions of this chapter can be summarized as follows:

- This is the first work to defend against privacy leakage attacks [86] in the DRL context. Specifically, we apply an exponential mechanism of DP to an agent’s

observations in training in order to prevent the environmental information from being recovered by a privacy leakage attacker. The defense method can effectively reduce recovery rate to the DRL environment from privacy leakage.

- The proposed method can dynamically adjust the privacy budget to guarantee privacy for both the agent and the environment, as well as the utility of the trained agent's policy.
- We design a comprehensive experiment to support our proposed method.

## 5.2 Related Work

### 5.2.1 Sensitivity of RL Training Environment

DRL policies are demonstrated to be highly sensitive to training hyper-parameters, including reward scale, environment dynamics and random seeds. Variation in these parameters can lead to significantly different results [48]. A DRL agent typically over-fits to the original training environment and its policy frequently cannot be generalized to unseen domains [114]; for example, a well-trained agent will still follow the same trajectory even once all obstacles in the grid have been removed after the training process [86], meaning that adversaries can attempt to steal information pertaining to the original training environment. These studies indicate that RL models have implicitly memorized the training environment and are thus vulnerable to attacks that steal private training environment information.

### 5.2.2 Membership Inference Attack

Only a small number of studies have explored privacy leakage in RL. One such study proposes a privacy leakage attack [86] to infer RL agent training transition dynamics from certain recovered candidates using a genetic algorithm and neural network. Similar to the privacy leakage attack in RL, some studies have shown that machine learning models can leak various types of sensitive information contained in the training data [95]. Membership inference is a common type of tracing attack used to determine whether or not a specific individual is a member of a given dataset [31]. Membership inference attacks on machine learning models were first proposed by [106], which trained multiple shadow policies and a neural network to identify whether a specific data point is contained in the training dataset. A similar idea is also applied to attack the differentially

private deep learning model, revealing that differentially private deep learning models are vulnerable to membership inference attack [95]. Some further studies related to membership inference attacks have examined why membership inference is possible [124] and mounted inference attacks on other forms of generative models [46]. A further work [108] shows how membership inference attacks can be used to determine if a model was trained using any individual user’s personal information. Another study [15] contends that the fact that membership inference is highly related to unintended memorization.

### **5.2.3 Inverse Reinforcement Learning**

Inverse RL was initially proposed in [81] as a paradigm to infer an expert RL agent’s reward function which is optimized from a given policy or observed agent behavior. The concept has attracted substantial interest in the communities of artificial intelligence, control theory, machine learning and psychology [5]. Inverse RL approaches include maximum margin approaches [2, 93] which maximally attain the experts’ performance even without recovering the reward function, and probabilistic approaches [64, 144, 145], which employ maximum entropy to resolve the ambiguity in choosing a distribution over decisions in the form of a maximum likelihood problem, and have also been further developed to incorporate adversarial learning [38]. Inverse RL is beneficial to some scenarios, such as re-optimizing a reward in novel environments [34], and can be used to infer a RL agent’s intentions [33].

### **5.2.4 Privacy Preservation in Reinforcement Learning**

Over the past decade, many studies have successfully combined DP and RL over the past decade. However, most of these studies applied RL algorithms to improve the performance of DP rather than guaranteeing the privacy of RL. RL is primarily applied dynamically adjust the allocation of privacy budgets in DP in many scenarios, such as privacy in trajectory, where RL is embedded into the DP mechanism to improve trajectory privacy protection in VANET [16] and dynamic data publishing [40]. Therefore, RL can benefit DP in the context of dynamic privacy budget adjustment.

Some studies have explored DP for RL in certain specific application scenarios. One study [7] focused on policy evaluation in the batch case by proposing regularized least-squares algorithms with output perturbation and bounding the excess risk due to the privacy constraints. Another work [128] provided a differentially private Q-learning

algorithm with function approximation to address the control problem with private rewards and public states. Several studies have applied DP in multi-agent RL systems to protect the process of knowledge transfer from an experienced agent to a less experienced agent. Two existing works utilized a similar idea, proposing a differentially private advising framework [104] and a differentially private knowledge transfer framework [18] respectively to perturb an experienced agent’s Q-table while transferring the knowledge. These two methods also used the concept of neighboring datasets to allow more possible knowledge transfer, though, two agents’ actions differ by one record. A similar work applied an exponential mechanism to help an experienced agent select a perturbed action for use as advice that aids it in avoiding malicious agents [137].

Far fewer works have addressed DP for general RL. One study [126] designed privacy-preserving exploration policies for episodic RL with joint DP [49] in which each user only receives their own sets of output, thereby providing privacy formulation with probable approximate correctness and regrets in RL. Another work [41] provided an alternative solution with a stronger privacy guarantee using the notion of local DP [55]. However, these two papers aimed to provide a privacy guarantee for the personal information of RL agent users, who may contribute training data collaboratively, rather than considering the training environment of the agent itself. One study [136] had a similar idea to ours with regard to obfuscating an RL agent’s training environment through the planning of logistic-like scenarios using the Laplace mechanism; however, this work still involves multiple users (called logistic centers in the paper) collaboratively contributing pieces of a training environment. Also, the work did not require a strong utility guarantee, because users could manually adjust the planning strategy rather than following the agent’s strategy. Therefore, we herein initiate the study of privacy preservation of the training environment of an RL agent in sequential decision-making, with a particular focus on defense against privacy leakage attacks in RL.

## 5.3 Preliminaries and Problem Definition

### 5.3.1 Privacy Leakage of RL

Privacy leakage attacks in the DRL context were first proposed in [86] aiming to infer the transition dynamics  $\mathcal{T}$  of an MDP using a well-trained policy  $\pi$  and other components of that MDP in a DRL setting. The attack is applicable to two scenarios with limited prior knowledge. In the first scenario, an attacker has access to prior knowledge of

some structural constraints related to the environment and then searches a transition dynamics  $\mathcal{T}$  maximally similar to the target transition dynamics  $\mathcal{T}_{target}$  using a genetic algorithm. The attacker maintains a population of transition dynamics candidates  $\{\mathcal{T}\}$  satisfying the known constraints in each searching iteration, and acquires the most similar transition dynamics  $\mathcal{T}$  by calculating how similar the action-space probability distribution of the induced optimal policy  $\pi_{\mathcal{T}}$  is to the  $\pi_{target}$ 's in order to update the population of  $\{\mathcal{T}\}$  in the next searching iteration. The second scenario is an application for membership inference attack in a DRL setting, in which the attacker trains shadow policies to identify the target transition dynamics from a set of transition dynamics candidates  $\{\mathcal{T}\}$ .

### 5.3.2 Problem Definition

A DRL agent will leak information about its environment during training, because the agent's optimal policy contains large amounts of learning experience and memories of environmental information. The root cause of privacy leakage in DRL is the agent's observations of the training environment. The primary task of a DRL agent is to train an optimal policy in the current environment to meet the requirements of the task. When an agent begins to learn its task in a new DRL environment, it takes actions based on what it observes from the environment at every state. The agent obtains rewards from selected actions and links these rewards to observations from the environment. Thus, the agent's observations can significantly affect the actions it takes, while the rewards it obtains further affect the policy. Through repeatedly observing and learning of the environment, the agent will take actions with increasing confidence and tends to remember the sequence of actions it takes alongside its memory of the environment. When the agent is brought into a new open environment, it will continue to repeat the sequence of actions from memory, resulting in the leaking of private information about its trajectory and therefore the original environment. Therefore, it is difficult for us to prevent the leakage of environmental information from an agent's well-trained policy.

Privacy leakage attacks are undoubtedly a great challenge to DRL because these attacks only require an agent's policy as input. Besides some prior knowledge of structural constraints related to the environment, only a well-trained policy from the agent is needed; subsequently, the fitness score can be used to calculate how similar an inferred transition dynamics candidate  $\mathcal{T}$  is to the well-trained policy. However, we contend that this very strength is also a fundamental weakness of the privacy leakage attack, as it is overwhelmingly dependent on the agent's well-trained policy. Specifically, the

attack performance is strongly related to the honesty of the agent during the training and the accuracy of the well-trained policy itself. In theory, an attacker can only infer information about a protected environment if an agent’s policy is trained in this protected environment; in other words, the recovery rate will decrease significantly if the agent’s policy is trained in an obfuscated environment. Although environmental information is objective and definite, we can obfuscate the agent’s observations in order to obfuscate the environmental information in the agent’s memory. Therefore, we suppose that the key to defense privacy leakage attack is to obfuscate a DRL agent’s observations during training while simultaneously guaranteeing the utility of the trained policy.

## 5.4 Differentially Private Deep Reinforcement Learning (DP-DRL)

### 5.4.1 Overview of Methodology

A DRL agent’s observations play a significant role in acting as a bridge to link the agent’s actions and environmental information. The agent’s observations at the current state are inputs of the neural network  $Q$  function, which guides the agent in taking an action along with previous learning experiences. Therefore, we believe that protecting the privacy of a DRL agent’s observation is the most straightforward and effective method. In this work, we accordingly propose a *Differentially Private Deep Reinforcement Learning (DP-DRL)* method to protect information about a DRL agent’s training environment against privacy leakage attack. We apply the exponential mechanism of DP to protect the agent’s observations from each visited state and further protect the trained policy. A brief overview of our proposed method is presented in **Figure 5.2**. Our method acts on the agent’s observations by obfuscating one observation element for each visited state based on the probability distribution of the outputs of the exponential mechanism. The agent’s DRL model receives obfuscated observations as inputs to implement action selection and model training; thus, the agent’s actions and trained policy are also protected by the randomization from DP. An attacker attempting a privacy leakage attack can only access a protected policy trained by obfuscated observations and will thus infer information about the obfuscated training environment rather than the true training environment. Our proposed method provides a privacy guarantee that an attacker cannot recover accurate environment information, thereby significantly reducing the recovery rate from a privacy leakage attack.



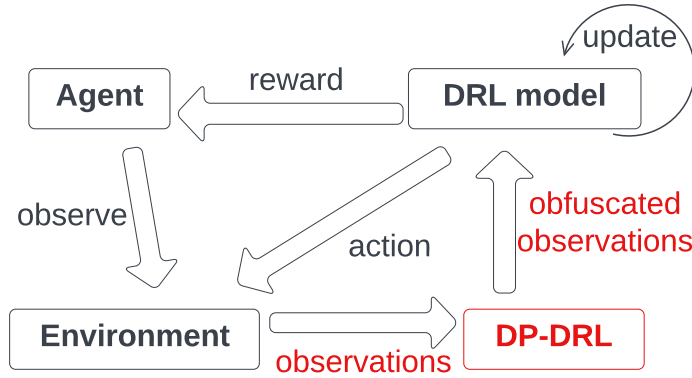


Figure 5.2: Overview of DP-DRL

We contend that the exponential mechanism is the best solution to guarantee privacy preservation and model utility at the same time due to its precision and flexibility. Unlike the Gaussian or Laplace mechanisms, which add numeric noise to all data records of a dataset, the exponential mechanism allows users to select only one data record based on the normalized probability distribution from outputs, where these outputs are driven by the score function and privacy budget. Therefore, the performance of the exponential mechanism is determined by the definition of the score function and the setting of the privacy budget. We define the score function as a measurement of the observation element that is least important to the model utility at each state. The least important observation can make the least impact on the outputs from the neural network  $Q$  function, and will thus be granted the highest score by the score function. The highest score also indicates the highest probability of being selected for obfuscation. We also propose a solution to dynamically adjust the privacy budget in order to balance the privacy guarantee and model utility to the greatest extent possible. In our method, the privacy budget is positively correlated to the difference between the minimal impact and the maximal impact on  $Q$  function. Further details are provided in the following subsections.

### 5.4.2 Definition of Score Function

As noted above, we set the score function as a measurement of the importance of all observation elements at the current state. Specifically, the least important observation element is awarded the highest score by the score function and has the highest probability of being selected for obfuscation. We can observe from **Algorithm 10** that an agent’s observations are inputs of the neural network, action-value function  $Q$ , which outputs

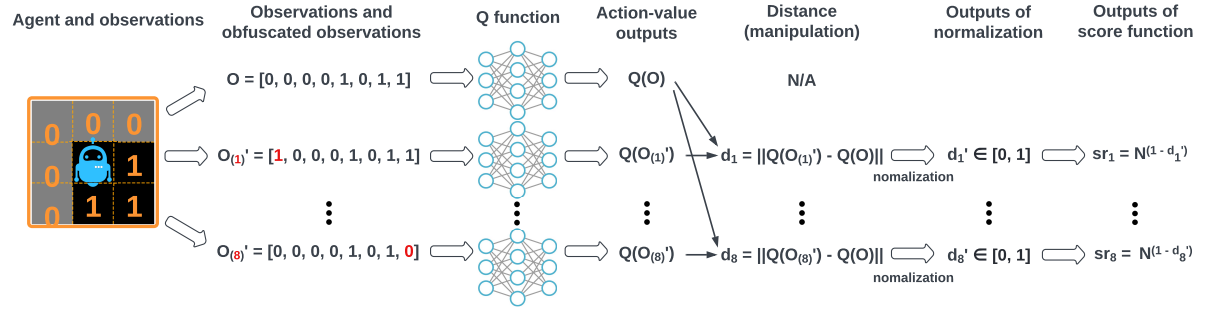


Figure 5.3: Process of the score function

an action-value vector. The action with the largest value in the action-value vector will be chosen by the agent under a greedy policy. A reward is returned from the selected action, and both the obtained reward and action are saved in transit as a learning experience that participates in the updating of the neural network's  $Q$  function by means of gradient descent and back-propagation. Therefore, the model utility is determined by the convergence of the  $Q$  function and outputs from the  $Q$  function based on observations as inputs.

---

**Algorithm 9** Score function

---

**Require:** A set of observations  $O = \{o_1, o_2, \dots, o_n\}$ ,  $Q$  function  $Q()$ , a positive constant  $N$ ;

- 1:  $D = \phi$ ;
  - 2:  $SR = \phi$ ;
  - 3: Compute  $Q(O)$ ;
  - 4: **for**  $i = 1$  to  $n$  **do**
  - 5:   Get  $O'_{(i)}$  by obfuscating the  $i^{th}$  element from  $O$ ;
  - 6:   Compute  $Q(O'_{(i)})$ ;
  - 7:    $d_i \leftarrow \|Q(O'_{(i)}) - Q(O)\|$ ;
  - 8:    $D \leftarrow D \cup d_i$ ;
  - 9: **end for**
  - 10:  $d_{min} = \min(D)$ ;
  - 11:  $d_{max} = \max(D)$ ;
  - 12: **for**  $i = 1$  to  $n$  **do**
  - 13:    $d'_i \leftarrow \frac{d_i - d_{min}}{d_{max} - d_{min}}$ ;
  - 14:    $sr_i \leftarrow N^{1-d'_i}$ ;
  - 15:    $SR \leftarrow SR \cup sr_i$ ;
  - 16: **end for**
  - 17: **return**  $SR, d_{min}, d_{max}$
- 

To explore the impact of our proposed method on model utility, we need to analyze the difference in the outputs of  $Q$  function between original observations and obfuscated

observations. We contend that the observation element which results in the smallest manipulation of the  $Q$  function’s outputs plays the least important role in action selection. Thus, the score function is related to the distance between  $Q$  function outputs with true observations as inputs and  $Q$  function outputs with obfuscated observations as inputs. The observation element with the smallest distance is considered to be the least important element, and is given a larger score that improves the likelihood of its being chosen to obfuscate. Therefore, there is an inverse monotonicity between an observation element’s score and its distance from the  $Q$  function.

**Figure 5.3** shows the definition of the score function. Assume a DRL agent can observe the closest eight grid squares at each state. For each grid square, 0 denotes a wall or obstacle while 1 indicates a free space. The agent obtains the set of true observations  $O$ , and brings  $O$  into the neural network  $Q$  function for the action-value outputs  $Q(O)$ . The agent then obfuscate one observation element, and obtains the obfuscated set of observations  $O(i)'$  with the obfuscation on the  $i^{th}$  element where  $i \in [1, n]$ . The agent brings these obfuscated observations into the  $Q$  function and outputs the action-value vectors  $Q(O(i)')$ . To measure the importance of observations elements, we calculate the distance  $d_i = \|Q(O(i)') - Q(O)\|$ , which indicates the manipulation from obfuscation on the  $Q$  function’s outputs. To simplify the calculation of the score function, we introduce a normalization function,  $d'_i = \frac{d_i - d_{min}}{d_{max} - d_{min}}$  to map  $d_i$ , which is theoretically in the range of  $[0, \infty)$ , to the new range of  $[0, 1]$ , where  $d_{min}$  and  $d_{max}$  are the smallest and largest distance, respectively. As mentioned above, there should be an inverse monotonicity between scores and distances, because the score function is set to measure the least importance while distance is positively correlated to importance. Thus, the output of the score function  $sr_i = N^{1-d'_i}$  where  $1 - d'_i$  denotes the least importance and  $N$  is a positive constant to scale the output. The algorithm of the score function is presented in **Algorithm 9**.

### 5.4.3 Dynamic Adjustment of Privacy Budget

The privacy budget is applied to adjust the level of randomization from DP, which can significantly affect the level of privacy preservation and model utility. There is a common perception that a smaller privacy budget results in a stronger privacy guarantee but a lower model utility due to the higher degree of randomization. A small privacy budget scales all scores from the score function down, which results in close outputs of all observation elements from the exponential mechanism and a similar probability of being chosen. Conversely, a large privacy budget leads to a weak privacy guarantee: scores are

scaled up, and the element with the highest score has a large probability of being chosen every time, meaning that the mechanism is insufficiently random and thus loses the ability to protect data privacy. The value of the privacy budget thus plays a significant role in the performance of applying DP. However, the question of how to determine the most beneficial value for the privacy budget has always been a challenge while applying DP. The most common method is to consider the privacy budget as a constant in a pre-defined range. The DP user tests variant privacy budgets within this pre-defined range according to the performance of both the privacy guarantee and model utility. However, this method can only test the performance of privacy guarantee and model utility from practical applications rather than providing a real balance between the two.

Therefore, we opt to dynamically adjust privacy budget at each step so as to dynamically balance the privacy guarantee and model utility to the greatest extent possible. We have explored the impact on outputs of the  $Q$  function from each observation element and calculated  $d_i$  as the distance of  $Q$  function outputs between obfuscated observations and true observations. On one hand, if selecting a different observation element results in a huge difference in the outputs of the  $Q$  function, this means the randomization may have a significant impact on model utility. In this case, we set a larger privacy budget to guarantee that the least important observation element is more likely to be selected, thereby reducing the manipulation of  $Q$  function outputs and minimizing the impact on model utility. On the other hand, if selecting different observation elements only produces a minor difference in the  $Q$  function outputs, the selection of obfuscated observation will not have too great an impact on model utility. Thus, we can provide a stronger privacy guarantee by setting a smaller privacy budget.

We can reuse the outputs of the  $Q$  function and computed distance  $d_i$ , which measures the distance of outputs from the  $Q$  function between true observations as inputs and obfuscated observations (with an element obfuscated) as inputs. We obtain the largest distance as  $d_{max}$  and the smallest score as  $d_{min}$ .  $|d_{max} - d_{min}|$  is the largest difference of  $Q$  function outputs among observation elements. When  $|d_{max} - d_{min}|$  is small, a small privacy budget is set to provide a stronger guarantee, while a larger privacy budget will be set to reduce the impact on model utility if  $|d_{max} - d_{min}|$  is large. Therefore, the privacy budget  $\epsilon$  is positively correlated to  $|d_{max} - d_{min}|$ . We define the privacy budget  $\epsilon = \arctan(d_{max} - d_{min})$ .

#### 5.4.4 Algorithm of DP-DRL

The algorithm of DP-DRL is presented in **Algorithm 10**. Our method obfuscates one element of the agent’s true observations based on a probability distribution and returns a privacy-preserved observation to the agent for further action selection and training. Assume  $\mathcal{O}(s)$  is the agent’s true observations on the state  $s$ , containing a set of observation elements  $\{o_1, o_2, \dots, o_n\}$  where  $n$  represents the length of observations. We apply an exponential mechanism of DP to protect the agent’s observations. The score function is presented in **Algorithm 9** as an essential component of the exponential mechanism that measures the importance for all observation elements in  $\mathcal{O}(s)$ . **Algorithm 9** returns a set of scores  $SR$  and both the smallest distance  $d_{min}$  and the largest  $d_{max}$ . The privacy budget is dynamically computed based on the equation  $\epsilon = \arctan(d_{max} - d_{min})$ . The sensitivity  $S$  is applied to measure the largest difference of score function outputs between two neighboring datasets. As the score function is designed to identify the observation that is least important to the  $Q$  function, the sensitivity  $S = sr_k - sr_m$ , where  $sr_k$  is the greatest element and  $sr_m$  is the second greatest element of the  $SR$ . We use  $\mathcal{E} = \{e_0, e_1, \dots, e_{n-1}\}$  to represent the exponential results for all observation elements. Because the exponential mechanism of DP aims to select one queried result based on the probability, but  $\mathcal{E}$  is a set of values instead of probabilities, a normalization function should be incorporated to calculate the probability distribution  $\mathcal{P} = \{p_0, p_1, \dots, p_{n-1}\}$  based on  $\mathcal{E}$ , where  $p_i = \frac{e_i}{\sum_{j=0}^{n-1} e_j}$ ,  $1 \leq i \leq n$ .  $\mathcal{O}(s)$  becomes  $\mathcal{O}^*(s)$ , with the element  $o_i$  that is chosen to be obfuscated denoted as  $o_i^*$  using a roulette wheel selection method based on  $\mathcal{P}$ . The method returns the obfuscated set of observations  $\mathcal{O}^*(s)$  to the agent for further action selection and training.

**Algorithm 10** DP-DRL

---

**Require:** Current state  $s$ , agent's observation set  $\mathcal{O}(s) = \{o_1, o_2, \dots, o_n\}$ ;

- 1:  $\mathcal{E} = \phi$ ;
  - 2:  $\mathcal{P} = \phi$ ;
  - 3:  $SR = \{sr_1, \dots, sr_n\}, d_{min}, d_{max} \leftarrow \mathbf{Algorithm\ 9}(\mathcal{O}(s))$ ;
  - 4:  $\epsilon \leftarrow \arctan(d_{max} - d_{min})$ ;
  - 5:  $S = sr_k - sr_m$ , where  $sr_k$  is the greatest element and  $sr_m$  is the second-greatest element of the  $SR$ ;
  - 6: **for**  $i = 1$  **to**  $n$  **do**
  - 7:      $e_i \leftarrow \exp(\frac{\epsilon \cdot sr_i}{2S})$ ;
  - 8:      $\mathcal{E} \leftarrow \mathcal{E} \cup e_i$ ;
  - 9: **end for**
  - 10: **for**  $i = 1$  **to**  $n$  **do**
  - 11:      $p_i \leftarrow \frac{e_i}{\sum_{j=0}^{n-1} e_j}$ , where  $e_i, e_j \in \mathcal{E}$ ;
  - 12:      $\mathcal{P} \leftarrow \mathcal{P} \cup p_i$ ;
  - 13: **end for**
  - 14: Randomly generate a probability  $p \in [0, 1]$ ;
  - 15:  $i \leftarrow \begin{cases} i, & \text{if } \sum_{j=0}^{i-1} p_j < p \leq \sum_{j=0}^i p_j, i > 0 \\ 0, & \text{if } 0 \leq p \leq p_0 \end{cases}$  ;
  - 16:  $o_i$  is chosen to be obfuscated as  $o_i^*$ ;
  - 17:  $\mathcal{O}^*(s) \leftarrow \mathcal{O}(s) \setminus o_i \cup o_i^*$ ;
  - 18: **return**  $\mathcal{O}^*(s)$
- 

**Algorithm 11** introduces how DP-DRL can be applied in a typical Deep Q-learning with experience replay algorithm. In each training epoch, the agent starts training from the state at the starting point, and terminates training when it has reached the destination or meets the failure criteria. The agent first observes the environmental information at the current state and transform observations as a vector or a set. The set of observations are input into DP-DRL, shown in **Algorithm 10**, and returned to agent as an obfuscated set of observations. A randomly generated probability  $p$  and a greedy policy threshold  $\epsilon_g$  can determine the method of action selection: the agent randomly selects an action when  $p > \epsilon_g$ , or selects an action based on the current action-value function  $Q$  with the observations vector and weights  $\theta$  otherwise. Here,  $\epsilon_g$  is usually set as a large percentage in the range  $[0, 1]$  (e.g., 0.9) or an automatically increasing function convergent to the large percentage. Once the agent selects an action, it needs to record a transition to capture its learning experience; this will include its obfuscated observations, the action taken, the reward obtained from the action taken, and observations made at the next state transited to from the current state by taking the action. The newly recorded transition will be incorporated into agent's memory pool, or replace another

transition according to a first-in first-out policy if the capacity is full. The agent samples a random mini-batch size of transitions from the pool and updates its action-value function  $Q$  using a gradient descent policy based on the current obtained reward in the sampled transitions and the maximum reward the agent can obtain from the next state in the sampled transitions. The agent’s target action-value function  $\hat{Q}$  will be duplicated from the updated action-value function  $Q$  every constant  $C$  steps, in order to guide better action selection.

---

**Algorithm 11** Deep Q-learning with DP-DRL

---

**Require:** Initialize replay memory  $D$  with capacity  $N$ , action-value function  $Q$  with random weights  $\theta$ , target action-value function  $\hat{Q}$  with weights  $\theta^- = \theta$ ;

```

1: for  $e = 1$  to  $EPOCHS$  do
2:   Reset state  $s$  as starting point;
3:   for step = 1 to  $STEPS$  do
4:     Get observations  $\mathcal{O}(s)$  at  $s$ ;
5:      $\mathcal{O}'(s) \leftarrow$  Algorithm 10( $\mathcal{O}(s)$ );
6:     Randomly get probability  $p, p \in [0, 1]$ ;
7:     if  $p > \epsilon_g$ , where  $\epsilon_g$  is threshold of greedy policy then
8:       Randomly select action  $a, a \in \mathcal{A}$ ;
9:     else
10:       $a \leftarrow \operatorname{argmax}_{\mathcal{A}} Q(\mathcal{O}'(s), \mathcal{A}; \theta)$ ;
11:    end if
12:    Obtained reward  $r \leftarrow \mathcal{R}(s, a)$ ;
13:    The next state  $s' \leftarrow \mathcal{T}(s, a)$ ;
14:    The transition  $t \leftarrow (\mathcal{O}'(s), a, r, \mathcal{O}'(s'))$ ;
15:     $D \leftarrow D \cup t$ ;
16:    Sample random mini-batch of transitions  $(\mathcal{O}'_j, a_j, r_j, \mathcal{O}'_{j+1})$  from  $D$ ;
17:    Set  $y_j \leftarrow \begin{cases} r_j, & \text{if next step } j+1 \text{ terminates the epoch} \\ r_j + \gamma \max_{\mathcal{A}} \hat{Q}(\mathcal{O}'_{j+1}, \mathcal{A}; \theta^-), & \text{otherwise} \end{cases}$ ;
18:    Gradient descent on  $(y_j - Q(\mathcal{O}'_j, \mathcal{A}; \theta))^2$  with  $\theta$ ;
19:    Reset  $\hat{Q} = Q$  every constant  $C$  steps;
20:     $s = s'$ ;
21:    if  $s = s^*$  or  $s$  meets failure criteria, where  $s^*$  is winning state then;
22:      break;
23:    end if
24:  end for
25: end for

```

---

### 5.4.5 Discussion

Our method provides a strong privacy guarantee by using the exponential mechanism of DP. As discussed above in the problem definition, the primary reason for privacy leakage in DRL is the agent’s observations. Therefore, our approach represents an effective and straightforward way to protect the agent’s observations directly. DP naturally benefits DP-DRL due to its strongly statistical privacy guarantee. DP can be involved in a wide range of fields due to its statistical properties and demonstrates strong flexibility in applications such as perturbing numeric data or securely selecting a precise data record. Our method benefits from the exponential mechanism of DP. The exponential mechanism can select one precise record based on probability distribution from the score function instead of perturbing all elements of the vector with numeric noise like the Gaussian mechanism or Laplace mechanism. When obfuscating the agent’s observations with DP-DRL, the agent can still observe the environment to the greatest extent under protection because obfuscated observations can be very similar to original observations with privacy preservation. Therefore, our proposed method can guarantee strong privacy and effective training at the same time.

Our method can maximally guarantee the utility of trained policy while also providing a privacy guarantee; in so doing, it benefits from the definition of the score function in DP-DRL, which significantly improves the performance of the method. The score function is defined to measure and identify the least important of the agent’s observation elements by computing the difference in  $Q$  function outputs between the agent’s true observations and every possible obfuscated observation obtained via our method. The score function can accordingly provide a clear view of the impact of each observation on the outputs of the  $Q$  function by quantifying the importance of each observation element. The outputs of the score function can thus effectively guide the exponential mechanism to select the least important element for obfuscation, thereby guaranteeing the model utility to the greatest extent. Therefore, the score function plays a significant role in both providing a privacy guarantee and maintaining model utility.

DP-DRL also benefits from the dynamically adjustable privacy budget, which enables it to balance the privacy guarantee and model utility in an effective way. The privacy budget is applied to control the level of randomization from DP. We link the level of randomization to the impact of all observation elements on the outputs of the  $Q$  function. Although the privacy budget cannot directly change the magnitude of the obfuscation’s impact on the  $Q$  function, it can adjust the level of randomization to determine the selection of the observation element. The privacy budget and the level of randomization



are positively correlated to the maximal difference of  $Q$  function outputs caused by the selection of observation elements to be obfuscated. The privacy budget is adjusted to primarily consider model utility by choosing the least important element when the selection can significantly impact the  $Q$  function and model utility. Moreover, it provides a strong privacy guarantee if observation element selection will have little impact. Therefore, the dynamically adjustable privacy budget can maximally balance the privacy guarantee and model utility during the agent’s training by controlling the randomization level impacting the selection of the observation element.

Our proposed method is easy to use in that the performance of the method is controlled by one parameter. As the function score of the exponential mechanism is defined, only two parameters — the sensitivity  $S$  and the privacy budget  $\epsilon$  — are involved in adjusting the performance of the mechanism. Moreover, because the sensitivity  $S$  is the maximum gap between the outputs of queries from two neighboring datasets and literally a constant, the performance is only controlled by the privacy budget  $\epsilon$ . Our method depends on an adequate value of  $\epsilon$  to balance the privacy guarantee and model utility, which is the same as other approaches that apply DP to deep learning. A very small epsilon results in similar outputs from the exponential mechanism  $\exp(\frac{\epsilon q(\mathcal{O}, o)}{2S})$  for all elements of  $\mathcal{O}$ , and close to equivalent probabilities after normalization of a given element being selected for obfuscation. Although the nearly random selection can provide a strong privacy guarantee, it leads to poor and slow convergence of the agent learning process and will further impact model utility. By contrast, a large epsilon can provide only a limited privacy guarantee. Therefore, the key to applying our method is to find a suitable value of privacy budget  $\epsilon$  in the exponential mechanism.

The proposed method requires only a small amount of computation and has a very short running time. Our proposed method targets an agent’s observations at each state rather than manipulating the DRL model and policy. When an agent transits to a state and begins to observe the environment, the proposed method is immediately applied to obfuscate the agent’s observations, which prevents the agent from bringing real observations into policy training and removes the associated risk of privacy leakage. The agent’s process of obtaining observations is considered the preparation process at each state before taking actions and model training. Thus, our method maintains the structure of the original DRL model procedure, without introducing any new iterations or adding extra computation into the calculation of model updates, and thereby guarantees that the time complexity of the original DRL model is not changed.

## 5.5 THEORETICAL PROOF ANALYSIS

### 5.5.1 Differential Privacy Analysis

To prove that DP-DRL can provide a privacy guarantee, we need to prove that the method satisfies the definition of DP and that the privacy budget  $\epsilon$  is used to control the level of randomization in the obfuscation of observations. Our method uses the exponential mechanism of DP to protect an agent’s observations through obfuscation. The obfuscated observations act as inputs for the agent’s DRL model in each state to determine the action taken by the agent. Therefore, to prove that our method satisfies  $\epsilon$ -DP, we need to prove that the exponential mechanism in obfuscated observations satisfies  $\epsilon$ -DP, and that an agent’s action of applying obfuscated observations satisfies DP, separately.

**Theorem 5.1.** *DP-DRL satisfies  $\epsilon$ -DP.*

**Proof.** The first step of proving that DP-DRL satisfies  $\epsilon$ -DP is to prove that the exponential mechanism in our method satisfies  $\epsilon$ -DP. Suppose that we have a set of observations  $O$ , and another set of observations  $O'$  with one observation element obfuscated from  $O$ . Obviously,  $O \oplus O' = 1$ , so  $O$  and  $O'$  are a pair of neighboring datasets according to **Definition 2.1**. Moreover, suppose that  $\mathcal{M}$  denotes the exponential mechanism, meaning that  $\mathcal{M}(O)$  denotes applying the exponential mechanism on the set  $O$ . For an element  $o_i$  in the set  $O$ , we have  $\mathcal{M}(O, o_i) = \exp(\frac{\epsilon q(O, o_i)}{2S})$  based on **Definition 2.8**. It is easy to prove [30] that  $\Pr[\mathcal{M}(O) = o] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(O') = o]$ , which satisfies **Equation 2.1** in **Definition 2.2**. The equation indicates that applying the exponential mechanism to our obfuscated observations is  $\epsilon$ -differentially private.

The next step is to prove the action taken by the agent,  $a$ , is also  $\epsilon$ -differentially private.

**Lemma 5.1** (Post-Processing [30]). *Let  $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \rightarrow R$  be a randomized algorithm that is  $\epsilon$ -differentially private. Let  $f : R \rightarrow R'$  be an arbitrary randomized mapping. Then,  $f \circ \mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \rightarrow R'$  is  $\epsilon$ -differentially private.*

We have proved that the exponential mechanism  $\mathcal{M}$  applied on observations  $\mathcal{O}$  in our method satisfies  $\epsilon$ -DP. In **algorithm 11**, the agent’s action selection function  $a \leftarrow \operatorname{argmax}_{\mathcal{A}} Q(\mathcal{O}(s), \mathcal{A}; \theta)$  where  $a \in \mathcal{A}$ , is a deterministic mapping  $f : \mathcal{O} \rightarrow \mathcal{A}$ . It has been proven that Lemma 5.1 also applies to a deterministic function  $f$ , because any randomized mapping can be decomposed into a convex combination of deterministic

functions [30]. Therefore, the agent’s action selection is also  $\epsilon$ -differentially private based on Lemma 5.1.  $\blacksquare$

### 5.5.2 Model Utility

To analyze the impact of our proposed method on DRL model utility, we need to explore how the obfuscated observations affect  $Q$  function, which is a neural network in DRL algorithm, and calculate whether there exists a bound that can limit the difference between the results of  $Q$  function from the original observations and the obfuscated observations. To do so, we introduce the concepts of Lipschitz continuity and the Lipschitz constant (Definition 5.1); here, the Lipschitz constant is the upper bound of the difference between the outputs of a function with two data points if the function is Lipschitz continuous. However, because not every neural network is Lipschitz continuous, we involve definition 5.2 to define a  $K$ -layer neural network composed by a set of fully connected layers and activation functions. It has been proven [127] that a MLP with a 1-Lipschitz activation function (e.g., ReLU) is Lipschitz continuous, and moreover that calculating the exact Lipschitz constant of the neural network is NP-hard; thus, we opt to estimate the Lipschitz constant of the neural network. A possible solution, AutoLip upper bound (Lemma 5.2), is proposed and proven in [127] to estimate the Lipschitz constant for an MLP-structured neural network.

**Definition 5.1** (Lipschitz continuity). A real-valued function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is called *Lipschitz continuous* if there exists a positive real constant  $L$  such that

$$(5.1) \quad \forall x_1, x_2 \in \mathbb{R}^n, \|f(x_1) - f(x_2)\|_2 \leq L \|x_1 - x_2\|_2$$

The smallest positive constant  $L$  for which this holds is called the *Lipschitz constant* of function  $f$ .

**Definition 5.2** (Multi-Layer Perceptron (MLP)). A  $K$ -layer *Multi-Layer Perceptron*  $f_{MLP} : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is the function

$$(5.2) \quad f_{MLP}(x) = W_K \circ \rho_{W_{-1}} \circ \cdots \circ \rho_1 \circ W_1(x)$$

where  $W_K : x \mapsto \omega_k x + b_k$  is an affine function and  $\rho_k : x \mapsto (g_k(x_i))_{i \in [1, n_k]}$  is a non-linear activation function.

**Lemma 5.2** (AutoLip upper bound [127]). *For any MLP with 1-Lipschitz activation functions (e.g., ReLU), the AutoLip upper bound on the Lipschitz constant is*

$$(5.3) \quad \hat{L}_{AL} = \prod_{k=1}^K \|W_k\|_2$$

**Theorem 5.2.** *If a DRL algorithm applies a neural network  $f(x)$  as  $Q$  function, where  $f(x) = W_K \circ \rho_{W_{-1}} \circ \dots \circ \rho_1 \circ W_1(x)$  is a  $K$ -layer MLP (definition 5.2) with 1-Lipschitz activation function (ReLU), for an agent's set of observations  $\mathcal{O}$  and obfuscated set of observations  $\mathcal{O}'$  obtained by DP-DRL, we always have*

$$(5.4) \quad \|f(\mathcal{O}) - f(\mathcal{O}')\|_2 \leq \prod_{k=1}^K \|W_k\|_2$$

**Proof.** Assume a neural network  $f(x)$  is the  $Q$  function of a DRL algorithm where  $f(x) = W_K \circ \rho_{W_{-1}} \circ \dots \circ \rho_1 \circ W_1(x)$  is a  $K$ -layer MLP. It is easy to prove that  $f(x)$  is Lipschitz continuous and that the estimated Lipschitz constant  $L_{f(x)}$  from Lemma 5.2 of  $f(x)$  is  $L_{f(x)} = \prod_{k=1}^K \|W_k\|_2$ . Assume that an agent's true set of observations is  $\mathcal{O}$ , and that the obfuscated set of observations obtained via our proposed method is  $\mathcal{O}'$ . According to the **Definition 5.1**, for any set of observations  $\mathcal{O}$  from agent, we have

$$(5.5) \quad \|f(\mathcal{O}) - f(\mathcal{O}')\|_2 \leq \prod_{k=1}^K \|W_k\|_2 \cdot \|\mathcal{O} - \mathcal{O}'\|_2$$

As DP-DRL only obfuscates one element of the observation set transmitting between 0 and 1, thus,  $\|\mathcal{O} - \mathcal{O}'\|_2 = 1$ . Thus, we have

$$(5.6) \quad \|f(\mathcal{O}) - f(\mathcal{O}')\|_2 \leq \prod_{k=1}^K \|W_k\|_2$$

which means  $\prod_{k=1}^K \|W_k\|_2$  is the upper bound of the neural network of  $Q$  function, and indicates how DP-DRL affects model utility. ■

### 5.5.3 Time Complexity Analysis

**Theorem 5.3.** *The time complexity of DP-DRL is  $O(n_e n_s)$ , where  $n_e$  is the number of training epochs and  $n_s$  is the training epoch length. DP-DRL does not increase the time complexity of the conventional DRL algorithm.*

**Proof.** To prove that DP-DRL does not change the time complexity of the conventional DRL algorithm, we first need to establish that the time complexity of the conventional DRL algorithm is  $O(n_e n_s)$  without our proposed method. By assuming the average running time of each block of lines of **algorithm 11** without DP-DRL (Line 5), the total running time  $T$  for the algorithm can be determined by summing all assumed average time  $T = t_{b1} + t_{b2} * n_e + t_{b3} * n_e * n_s$ , in which  $t_{b1}$ ,  $t_{b2}$  and  $t_{b3}$  are the total average running time of initialization, Line 1 to Line 2 and Line 3 to Line 22 without Line 5, separately. Because  $t_{b1}, t_{b2}, t_{b3} \ll n_e, n_s$ , the equation above is turned into  $T = n_e * n_s$ . Therefore, the time complexity of DRL algorithm is  $O(n_e n_s)$ .

The next step is to prove that the time complexity of DP-DRL in **Algorithm 10** does not increase the time complexity of the conventional DRL algorithm. Assume the average running time for each block of lines of **Algorithm 9** and **Algorithm 10**. We firstly compute the total running time  $T_{SF}$  of **Algorithm 9** by summing all assumed average time  $T_{SF} = t'_{b1} + t'_{b2} * n + t'_{b3} * n + t'_{b4}$ , in which  $t'_{b1}$ ,  $t'_{b2}$ ,  $t'_{b3}$  and  $t'_{b4}$  are the total average running time of **Algorithm 9** (Line 1 to Line 5, Line 6 to Line 9, Line 10 to Line 13, and Line 14 to Line 18 separately), and  $n$  is the size of observations. Because  $t'_{b1}, t'_{b2}, t'_{b3}, t'_{b4} \ll n$ , so the equation above is turned into  $T_{SF} = 2n$ . Therefore, the time complexity of **Algorithm 9** is  $O(n)$ . It is obvious that the total average running time of **Algorithm 10**  $T_{DP-DRL} = 4n$  and its time complexity is also  $O(n)$ .

The final step is to compute the average running time of **Algorithm 11** with **Algorithm 10** applied,  $T^* = t_{b1} + [t_{b2} + (t_{b3} + T_{DP-DRL}) * n_s] * n_e$ , while turning into  $T^* = (1 + 4n) * n_e * n_s$ . Because  $n$  is the size of a set of observations and is a constant, and  $n \ll n_e, n_s$ , the time complexity of **Algorithm 11** is  $O(n_e * n_s)$ . At this point, we have proven that DP-DRL does not increase the time complexity of the conventional DRL algorithm. The experimental environment is established in a grid world so that a DRL agent can navigate the path. ■

## 5.6 Experiments

In this section, we present our experimental environment settings and experimental results. In order to demonstrate our method intuitively, we reproduce experimental environment and settings of a privacy leakage attack [86], then apply our proposed defense method in the same environment to observe the performance of DP-DRL against privacy leakage attack.

### 5.6.1 Experimental Environment

#### 5.6.1.1 Grid World

Grid World is a popular environment for setting up navigation problems to test RL algorithms. In this environment, an RL agent trains an algorithm to find the path for assigned tasks, such as by finding the shortest path between two points or reaching dynamic targets like as in the game Snake. A grid is composed of boundary walls, free space, obstacles, and some optional targets if the task requires. At each step, the agent is

normally able to take one of five possible actions, including "up", "down", "left", "right" and "stay"; moreover, the agent cannot penetrate through walls and obstacles. The agent can obtain positive rewards by reaching the end point and will be penalized if it collides with walls and obstacles or stays in one place. No rewards or penalties will be applied if the agent successfully passes to the next free space. The goal of the agent is to maximize the obtainable rewards in each training epoch.

### 5.6.1.2 Structural constraints

The grid is subject to structural constraints, which also constitute the attacker's prior knowledge (as discussed in the privacy leakage attack paper). First, there should be one and only one pair of starting and ending grid squares located in free space, the positions of which can be considered known to an attacker who can observe the agent's behavior. Second, any two free grid spaces are connected, which means the shortest path between any two free grid squares is definite. Third, the experiment is simulating a floor plan, so the thickness of walls and obstacles can only be 1; in other words, any  $2 \times 2$  grid cannot contain four grid squares of walls and obstacles.

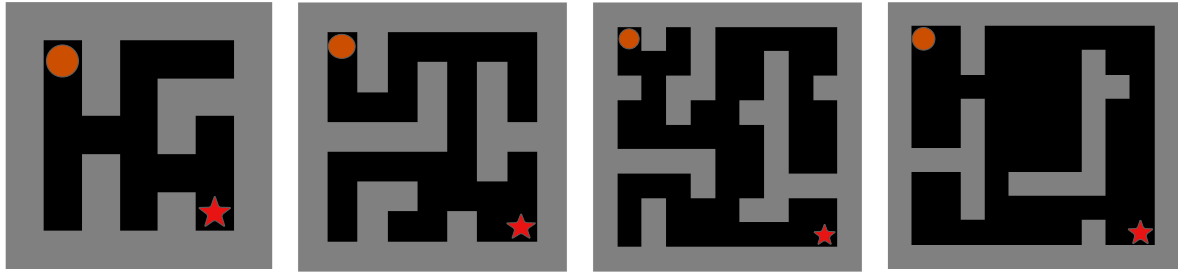
### 5.6.1.3 Agent's observations

Agent's observations are inputs for the DRL model that can significantly affect the trained policy. To reproduce the environmental settings of a privacy leakage attack, we assume that the agent takes the view of the closest 8 grids around in 8 directions (up, down, left, right, upper left, upper right, bottom left, and bottom right) in the current state. This is the most common view of DRL agents in navigation problems without human intervention, and is in line with real robotics cases such as sweeping robots.

## 5.6.2 Experimental Settings

### 5.6.2.1 Map Size

To enhance the randomness of the experiment and better prove the effectiveness of our method, our experiment is run within several randomly built maps with different sizes:  $7 \times 7$ ,  $9 \times 9$ , and  $11 \times 11$ . Maps are created under the above-mentioned structural constraints and the attacker's prior knowledge. Meanwhile, to provide an intuitive comparison of attack success rate, we also use the same  $12 \times 11$  map presented as the example map in the privacy leakage attack paper. Map examples are shown in **Figure 5.4**. These map sizes are selected because a smaller map may be too easy for the attacker,



(a) Example map of size  $7 \times 7$     (b) Example map of size  $9 \times 9$     (c) Example map of size  $9 \times 9$     (d) Example map of size  $12 \times 11$

Figure 5.4: Example maps in different sizes

who may perform well on a privacy leakage attack even by randomly guessing. On the other hand, a larger map requires a very long training time and increased computation capability, which may lead to underfitting on the part of the agent and subsequently to inaccurate policy. Therefore, we comprehensively determine that our selection of map size is persuasive and enables us to better demonstrate our defense method against privacy leakage attack.

### 5.6.2.2 Training Parameter

In this experiment, all policies, including the target policy and policies trained from a set of grid maps, are trained using DQN [78]. The input of the first linear layer is 8, as this is the size of the agent’s observations (the closest eight grid squares in eight directions). The output of the last linear layer is 5, as this is the size of the agent’s action spaces. The reward function plays a significant role in the RL environment by guiding the agent in searching for the optimal policy. Under our settings, the reward function is defined as 1 when the agent reaches the winning state successfully, -0.1 if the agent collides with obstacles or chooses to stay in place, and 0 if the agent passes to the next free space. The number of training epochs is set to 100,000, while the maximum number of steps in each training epoch is set to 500; thus, the agent is limited to a maximum of 500 steps in each epoch. This limit helps prevent the agent from infinitely repeating certain actions and accumulating ineffective training experience, as well as compressing training resources and time.

### 5.6.2.3 Parameters of DP-DRL

We also need to configure the parameters of DP-DRL to effectively implement the defense. Our defense method applies the exponential mechanism of DP, which requires the setting of the score function, the sensitivity, and the privacy budget. We have defined the score function in **Algorithm 9** determining the least important observation elements, i.e., those that minimally impact the outputs of the  $Q$  function. Most outputs are computed based on the DRL model in **Algorithm 9**, and we only need to control a constant  $N$  to scale the outputs in order to easily control the effect of the outputs and privacy budget.  $N$  is set to 10 in this experiment. The sensitivity of DP is the maximum difference to the outputs of the score function, which is computed based on the gap between the greatest score and the second-greatest score from the score function. The privacy budget is used to control the randomization level of the privacy preservation and strike a balance between the privacy guarantee and model utility. As discussed in the methodology section, we developed a method to dynamically adjust the privacy budget in order to dynamically balance the privacy guarantee and model utility. To explore how a dynamically adjustable privacy budget improves the performance of DP, we set a variant privacy budget that ranges from 0.1 to 1 in increments of 0.1 and that can be dynamically adjusted.

### 5.6.3 Performance Metrics

Recovery rate plays a significant role in measuring the performance of either a privacy leakage attack or a defense against such an attack; an attack that can maximally infer the training data and environment will have a high recovery rate, while a good defense method will significantly decrease this attack recovery rate. We apply multiple metrics to demonstrate our results efficiently. *General recovery rate* (GRR) is used to compare the similarity of all grid squares of the inferred and original maps. However, as mentioned above, the attack is based on the attacker’s prior constructional knowledge about the map (for example, that the outermost grid squares are walls), meaning that the attacker only needs to make inferences about the other squares. Therefore, we introduce *net recovery rate* (NRR) to measure the similarity of grid squares purely inferred by an attacker without taking prior knowledge into account. Both metrics are used to measure the recovery rate from privacy leakage attack.

As a defense method, we also need to guarantee the utility of the trained policy while protecting its private information from being leaked. In the DRL environment, we determine that the two most significant utility metrics are the *convergence rate* and



*obtained reward*. We compare the performance of the policies trained with and without the defense method, along with the rewards obtained in training epochs. Moreover, we also compare the number of epochs required for the agent to find the shortest walk path and the number of epochs in which the agent obtained the maximum expected reward. It should be noted here that in a given epoch, obtaining the maximum reward is not necessarily equivalent to finding the shortest path, as an agent may pass through a free space and receive 0 reward for taking a step.

### 5.6.4 Experimental Results

The experimental results indicate that our method can effectively defend against privacy leakage attack in the DRL context, as well as guaranteeing the utility of the model. In our defense method, the smaller the privacy budget applied, the lower the inference recovery rate of the privacy leakage attack, in other words, the better performance of defense.

#### 5.6.4.1 Attack benchmark

**TABLE 5.1** demonstrates the defense performance with no defense method, a variant fixed privacy budget and a dynamically adjustable privacy budget. The benchmark for our experiment is the GRR and NRR under a pure privacy leakage attack scenario with no defense method applied. From the benchmark, we can see that the recovery rate of privacy leakage attack is higher with smaller map sizes. There are three key reasons for this result. First, an attacker is required to recover fewer grid squares with larger weights of recovery rate in small-size maps, which primarily drives a higher recovery rate. Second, the attacker’s prior knowledge plays a significant role in the recovery process. When an attacker is confident regarding certain grid squares of a map, other squares around these squares are very likely to be identified through the application of knowledge regarding structural constraints; for example, if three grid squares of a 2 x 2 grid are identified as obstacles, the fourth square must be free space. Third, a training agent is more likely to fully traverse all grids in small-size maps with limited resources, which can cause significantly more environmental information to leak from the trained policy if the agent repeatedly visits the same state and takes the same actions.

By contrast, the privacy leakage attack recovery rate decreases with increasing map size. On one hand, the effectiveness of the attacker’s prior knowledge weakens due to the lower weight assigned to each recovered grid square. Due to the limited prior knowledge

	Map size $7 \times 7$		Map size $9 \times 9$		Map size $11 \times 11$		Map size $12 \times 11$	
Defense	GRR	NRR	GRR	NRR	GRR	NRR	GRR	NRR
No defense	94.26%	88.74%	87.29%	78.99%	85.20%	77.89%	84.17%	76.78%
$\epsilon = 1.0$	92.92%	86.12%	85.10%	75.36%	83.87%	75.90%	82.89%	74.91%
$\epsilon = 0.7$	89.39%	79.20%	83.77%	73.17%	81.13%	71.81%	79.92%	70.55%
$\epsilon = 0.4$	87.29%	75.09%	81.31%	69.10%	77.75%	66.76%	76.88%	66.09%
$\epsilon = 0.1$	84.82%	70.25%	80.11%	67.12%	76.65%	65.12%	76.19%	65.07%
<b>Dynamic <math>\epsilon</math></b>	<b>85.19%</b>	<b>70.97%</b>	<b>80.92%</b>	<b>68.46%</b>	<b>77.27%</b>	<b>66.05%</b>	<b>76.63%</b>	<b>65.72%</b>

Table 5.1: Defense Performance With Varying Privacy Budgets At Different Map Sizes

compared with the larger number of grid squares involved, an attacker is required to infer more grid squares through the agent’s policy without much confidence. On the other hand, a DRL agent tends to reuse experiences with higher rewards to optimize its policy rather than randomly traversing all grid squares of maps. The agent may pass only rarely over grid squares that are not in the critical path meaning that it will have less learning experience with these squares; as a result, the agent’s trained policy does not comprehensively reflect the environmental information, which increases the difficulty of recovery from the attack. Nevertheless, the attack still reaches a desirable recovery rate, as NRR is over 75% for all sizes of maps. Therefore, we deem this benchmark effective for assessing the performance of the proposed defense method in the experiment.

#### 5.6.4.2 Defense performance

**TABLE 5.1** also indicates that our proposed method can effectively defense privacy leakage attack, reducing the attack’s recovery rate by 10-15%. The attacker’s prior knowledge and uncertainty still play a role in the recovery process, meaning that our proposed defense method cannot infinitely reduce the attack accuracy. Thus, we believe that our proposed method meets expectations in a reasonable range. Apparently, the agent’s obfuscated observations can further obfuscate trained policy to prevent its learning experiences from the environmental information being disclosed. The obfuscated information prevents a privacy leakage attacker from accurately recovering the environmental information. The agent’s obfuscated observations can mislead an agent into remembering inaccurate environment information when optimizing its policy; thus, an agent’s trained policy can confuse an attacker by reflecting obfuscated information instead of the ground truth. **TABLE 5.2** presents a comparison of recovered maps between ground truth, no defense and our proposed defense method with  $\epsilon = 0.1$  and dynamically adjustable  $\epsilon$ . As



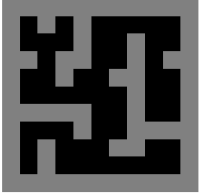
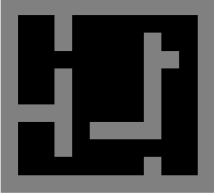
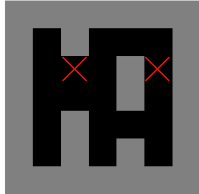
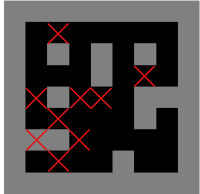
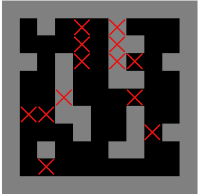
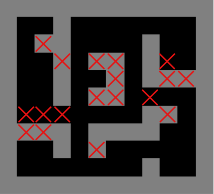
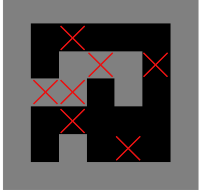
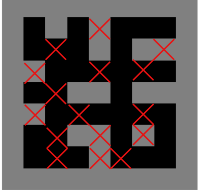

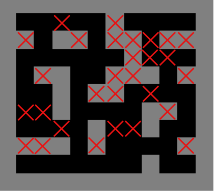
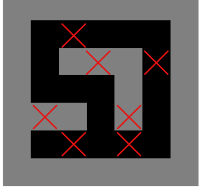
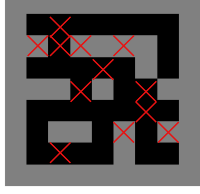
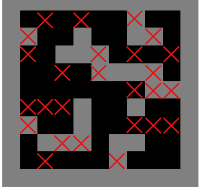
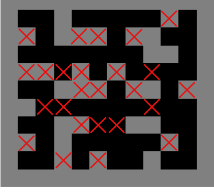
Method	Grid Maps			
	Grid size $7 \times 7$	Grid size $9 \times 9$	Grid size $11 \times 11$	Grid size $12 \times 11$
<b>Ground truth</b>				
<b>No defense</b>				
<b><math>\epsilon = 0.1</math></b>				
<b>Dynamic <math>\epsilon</math></b>				

Table 5.2: Example of Recovered Grid Maps

is evident, the maps recovered when no defense method is applied are similar to the ground truth; however, our proposed method significantly misleads the attacker, leading to recovered maps that differ significantly from the ground truth as the number of grid squares increases.

We further demonstrate how the privacy budget affects the defense performance of our proposed method. Our proposed method presents better performance when the privacy budget epsilon  $\epsilon$  grows smaller. Comparing variant fixed values of  $\epsilon$  in **TABLE 5.1** from 1.0 to 0.1, the recovery rate continuously decreases by up to 10%. Scores from the score function are deterministic based on the lowest importance of the observation element. The privacy budget  $\epsilon$  can control the level of randomization by scaling the effectiveness of scores mapping to outputs of the exponential mechanism and probability

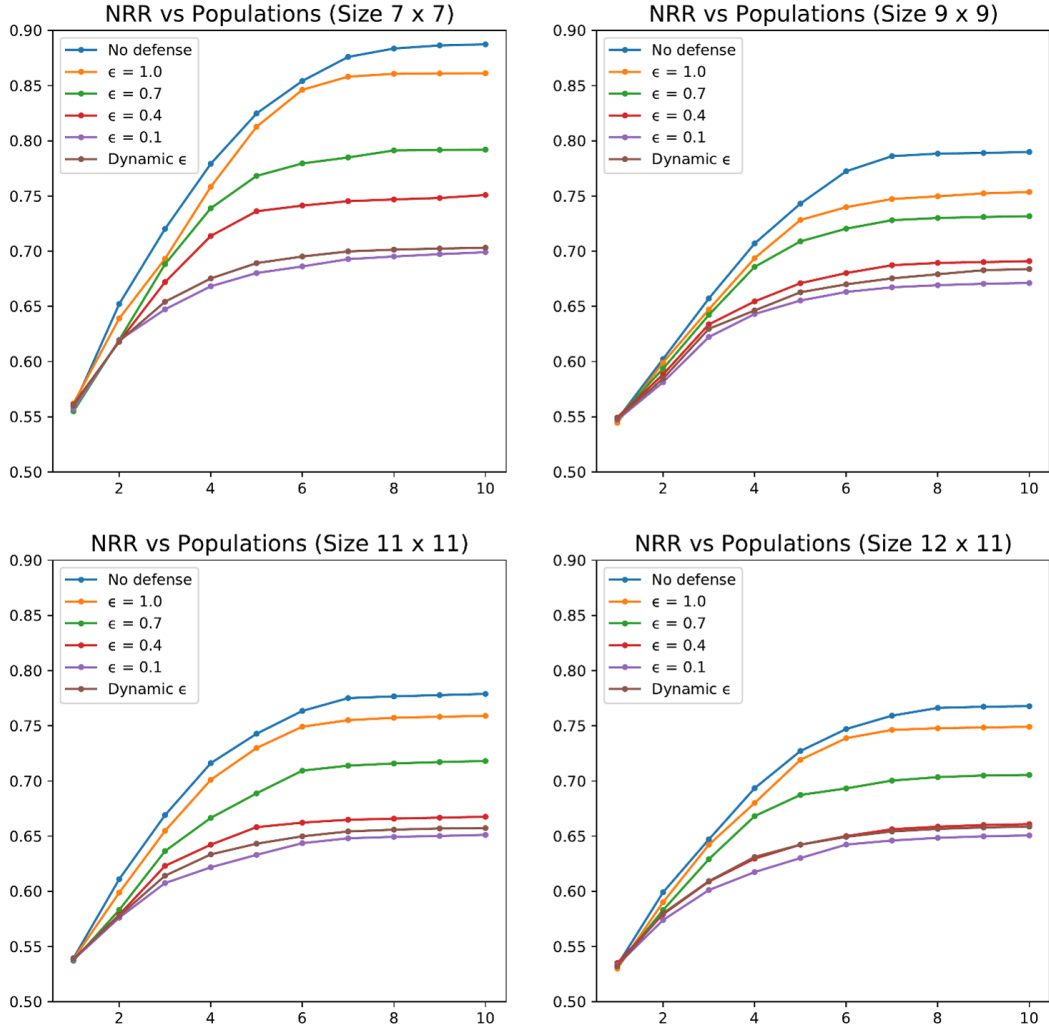


Figure 5.5: NRR vs populations at varying privacy budgets

distribution. A small  $\epsilon$  results in similar probabilities, or even equivalent probabilities if  $\epsilon$  is small enough, for all observation elements being selected for obfuscation. The small  $\epsilon$  undoubtedly provides a stronger privacy guarantee for the agent’s observations of the environmental information and further protects the privacy of the trained policy. Meanwhile, the dynamic privacy budget can provide an impressive privacy guarantee for information learned by the agent against privacy leakage attack. The performance of the dynamic privacy budget is in between the fixed privacy budgets of 0.4 and 0.1, but very close to 0.1. **Figure 5.5** plots the NRR of varying privacy budgets in genetic algorithm populations under privacy leakage attack. All curves have a similar shape and trend, however, the curve converges to a lower NRR with the smaller epsilon and

dynamic epsilons because the attack is convergent to a recovered map that differs more significantly from the ground truth. Overall, DP-DRL can effectively protect the privacy of environmental information against privacy leakage attack in DRL. When the privacy budget is fixed, the defense performance increases as the privacy budget is reduced. Our dynamic privacy budget also provides a very strong privacy guarantee, similar to when  $\epsilon = 0.1$ .

### 5.6.4.3 Utility

The experimental results also indicate that our proposed method can guarantee the utility of an agent’s trained policy as well as providing the privacy guarantee. The primary goal of an agent in a DRL environment is to effectively train the policy and accurately complete the task. In other words, an ideal agent is expected to learn how to obtain the maximum reward as fast, and also as steadily, as possible. Thus, it is necessary to explore how the defense method affects policy utility. In **TABLE 5.3**, we demonstrate the average obtained reward and average searching steps in the initial stage (300-1000 epochs) and final stage (last 1000 epochs) separately with varying privacy budgets. We found that agent randomly reached the destination for the first time between 100 and 250 epochs without any rules for all privacy budget settings. Once the agent obtained the positive reward, it displayed a tendency to reuse the experience and continuously converge to reach the maximum reward. Therefore, we contend that it is reasonable to measure the convergence speed of the initial stage after 300 epochs to exclude randomness from the agent itself.

Privacy budget	Initial Stage (300 - 1000 epochs)		Final Stage (Last 1000 epochs)		Time Overhead (s)	
	Min / Average reward / Max	Max / Average steps / Min	Average reward / Max	Average steps / Min	DL-DRL (each step)	DRL (each step)
No defense	-50 / -0.96 / 1	500 / 74.63 / 17	0.98 / 1	20.52 / 17	N/A	9.26e-04
$\epsilon = 1.0$	-50 / -1.13 / 1	500 / 87.35 / 17	0.97 / 1	22.84 / 17	1.63e-05	9.27e-04
$\epsilon = 0.7$	-50 / -1.42 / 1	500 / 97.17 / 17	0.95 / 1	24.92 / 17	1.62e-05	9.26e-04
$\epsilon = 0.4$	-50 / -1.49 / 1	500 / 105.42 / 17	0.94 / 1	26.16 / 17	1.62e-05	9.26e-04
$\epsilon = 0.1$	-50 / -1.60 / 1	500 / 111.91 / 17	0.92 / 1	28.28 / 17	1.63e-05	9.26e-04
<b>Dynamic <math>\epsilon</math></b>	-50 / <b>-1.18</b> / 1	500 / <b>90.22</b> / 17	<b>0.96</b> / 1	<b>23.13</b> / 17	1.64e-05	9.27e-04

Table 5.3: Model Utility Of Map Size  $12 \times 11$  At Different Privacy Budgets

For fixed privacy budget settings, the agent converges to the maximum reward more slowly when the privacy budget is reduced. In the initial stage of **TABLE 5.3**, a lower average reward and more average steps come along with the smaller privacy budget;

however, there is no significant difference on both metrics, especially when compared with theoretically maximum and minimum rewards and steps. It is worth noting that the agent with a dynamic privacy budget achieves very good convergence speed performance in the initial stage, highly similar to that when  $\epsilon = 1.0$ . This obviously indicates that our proposed dynamic privacy budget can effectively balance the trade-off between privacy preservation and agent convergence speed in the initial stage of learning.

**TABLE 5.3** also shows that our proposed method can guarantee the utility of the agent’s policy. In the final stage of the agent’s learning from the environment, the agent’s policy is sufficiently convergent to the maximally expected reward. Moreover, under considering fixed privacy budget settings, the proposed method performs very similarly to no defense method in terms of policy utility, average reward and average steps in the final stage. Especially when  $\epsilon = 1$ , the average reward only drops by less than 1%. Although it still follows the same trend of a lower epsilon bringing about a slightly worse performance, the drop is only within the range of 1% to 5%.

Better still, the dynamic privacy budget has only a very minor impact on the utility of the agent’s policy. Again, the agent with dynamic privacy budget has a very similar performance to when  $\epsilon = 1$ , in that the average reward reaches 0.96 out of 1, which is a drop of only about 1% compared to no defense method. Unlike the fixed privacy budget, which cannot effectively balance privacy guarantee and model utility at the same time, the dynamically adjustable privacy budget achieves impressive performance in terms of both providing a privacy guarantee and maintaining the utility of the agent’s policy. In summary, our defense method can effectively guarantee the utility of DRL policy in terms of both convergence speed and task performance.

#### 5.6.4.4 Time Overhead

The experimental results also demonstrate that our method has an extremely low time overhead. We have proven that our DP-DRL does not change the time complexity of the conventional DRL algorithm. The experimental results also confirm this proof. Specifically, we measure the average time overhead of applying our algorithm and the average total time overhead of DRL training each step of state transition in the experiment for both fixed and dynamic privacy budgets. The results are presented in **TABLE 5.3**. The average time overhead associated with executing observation obfuscation at each step of state transition is around  $1.62e-05$  s with our experiment and equipment settings, which only occupies around 1.7% of the average total time overhead of each step ( $9.26e-04$  s). The time overhead of both observation obfuscation and the total DRL training process

are the same under a varying privacy budget. The computation of the dynamic privacy budget barely changes the time overhead, which indicates that the time overhead of our method is not affected by the value of the privacy budget  $\epsilon$  or even the dynamically adjustable privacy budget. In general, our method has a very short time overhead compared to the conventional DRL training process.

## 5.7 Summary and Future Work

In conclusion, we have proposed a defense method named DP-DRL to defend against privacy leakage attacks in a DRL environment, which is the first work to defend against such attacks. We apply the exponential mechanism of DP to obfuscate the DRL agent's observations based on the probability distribution of the score function, which is defined to measure the least important of the agent's observation elements with minimal impact on  $Q$  function at each step, in order to protect the privacy of the agent's learning environment and trained policy. To maximally balance the trade-off between privacy guarantee and the utility of task performance, we designed a dynamically adjustable privacy budget solution based on the magnitude of the observation elements' impact on  $Q$  function. The experimental results indicate that our defense method can effectively decrease the privacy leakage attack recovery rate while still maintaining the utility of the agent's policy. For future work, current privacy leakage attack methods rely on attackers possessing prior knowledge, such as the structural constraints of the environment and the agent's policy parameters. It would be a very interesting future research direction that exploring how an attacker might recover a map using only agent behaviors, and without any prior knowledge. Moreover, more defense methods should be proposed to guarantee the privacy of DRL.





## ONE PARAMETER DEFENSE - DEFENDING AGAINST DATA INFERENCE ATTACKS VIA DIFFERENTIAL PRIVACY

### 6.1 Introduction

In the last chapter, we have introduced our defense method, DP-DRL, to defend against privacy leakage attacks in a DRL context. The exponential mechanism of DP is well-suited for privacy preservation in the DRL algorithm with a neural network by obfuscating its inputs. Given the rapid development of neural networks and increasing application scenarios, we wonder if our approach can be more generalized to be applied to more applicable fields.

We find that a similar approach can also be applied to classification problems in the deep learning context. To the best of my knowledge, machine learning models are vulnerable to membership inference and model inversion attacks. In these types of breaches, an adversary attempts to infer a data record's membership in a dataset or even reconstruct this data record using a confidence score vector predicted by the target model. However, most existing defense methods only protect against membership inference attacks. Methods that can combat both types of attacks require a new model to be trained, which may not be time-efficient. In this chapter, we propose a differentially private defense method that handles both types of attacks in a time-efficient manner by tuning only one parameter, the privacy budget. The central idea is to modify and normalize the confidence score vectors with a DP mechanism which preserves privacy and

obscures membership and reconstructed data. Moreover, this method can guarantee the order of scores in the vector to avoid any loss in classification accuracy. The experimental results show the method to be an effective and timely defense against both membership inference and model inversion attacks with no reduction in accuracy.

On the back of the massive amounts of data we humans generate every day, machine learning (ML) has become a key part of many real-world applications, ranging from image classification to speech recognition [98]. However, these data often contain sensitive personal information that is vulnerable to a range of adversarial activities, including membership inference attacks [80, 106] and model inversion attacks [37, 133]. Both fall into the category of data inference attacks, which are launched by exploiting redundant information contained in confidence score vectors. For example, a machine learning model will usually be more confident in its prediction about a data record that is in its training dataset, over another data record that is not. Attackers can exploit this difference in confidence to determine whether a given data record is or not a member of the target model’s training dataset. Here, a confidence score vector is a probability distribution over the possible classes predicted by an ML model. Each score in the vector indicates the model’s confidence in a prediction of the corresponding class. The class with the largest confidence is predicted as the label of the input data record.

Data inference attacks can result in severe privacy violations. For example, consider a model that has been trained on data collected from people with a certain disease. If a particular individual’s data are known to be in the training dataset, the adversary can immediately infer that person’s health status. Another example is model inversion attack. As shown in **Figure 6.1**, an adversary can train an attack model to accurately reconstruct an input data record using only the confidence score vector, even if the data record is never seen by the attack model. Hence, defending against data inference attacks has been the focus of much attention in the privacy community.

The methods proposed to date can be roughly classified into four categories based on the defense techniques employed. Those in the first category use regularization techniques to reduce overfitting, such as  $L_2$  regularizer [106], dropout [98], model-stacking [98] and min-max regularization [79]. This is because overfitting is one of the major factors leading to the distinguish ability between member and non-member data records [106]. The shortcoming of these methods, however, is that distinguish ability, i.e., a model’s vulnerability, is not reduced directly. Moreover, they require retraining the target model which may not be very efficient for complex neural networks.

The second category of the methods is based on adversarial examples [53]. These

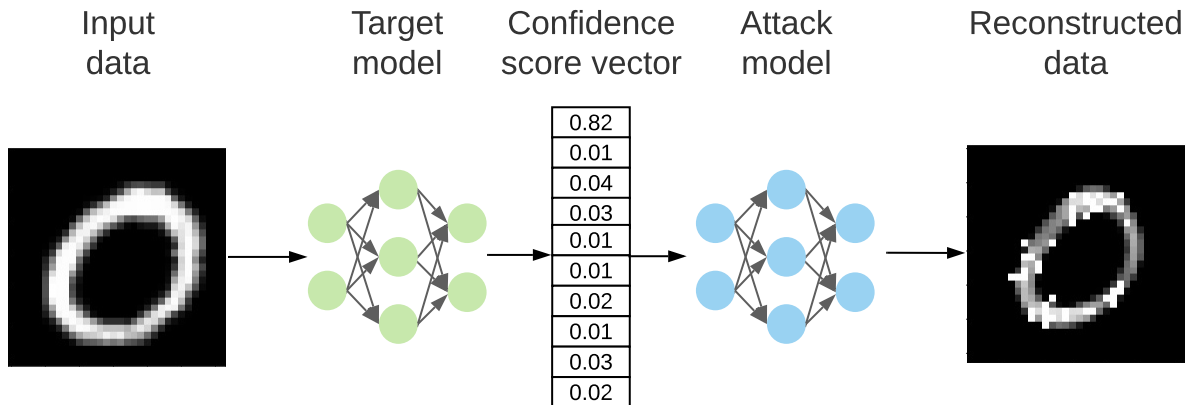


Figure 6.1: An input data record is accurately reconstructed by an attack model which has never seen this data record.

methods add a carefully crafted noise vector to a confidence score vector turning it into an adversarial example to mislead the attacker’s classifier. These methods have formal utility loss guarantees of confidence score vectors. However, their effectiveness depends on the transfer ability of adversarial examples[88], which might not generally reduce the distinguish ability of prediction scores [132].

The third category of the methods is based on deep neural networks [132]. These methods interpret defense goals as loss functions and train the deep neural networks as defense models to defend against attacks. These methods can minimize the content of information attackers can use to infer membership or reconstruct the data records. However, this strategy requires training new models which may not be very time-efficient.

The fourth category describes the DP methods [1]. These methods carry a theoretical guarantee of privacy. However, as DP is usually used during the training process, e.g., adding noise to gradients, there is always a large classification accuracy loss [79]. As shown by Jayaraman and Evans [51], existing differentially private machine learning methods rarely offer acceptable privacy-utility trade-offs for complex models. Moreover, since existing methods have to be integrated into the training of target models, they are not applicable to those target models which have already been deployed.

A common limitation shared by the existing defense methods is that all but one only protect against membership inference attacks. The only method to handle both membership inference and model inversion attacks was proposed by Yang et al. [132]. Their method, however, requires training new models which may not be very time-efficient. In this chapter, we develop a time-efficient defense method against both membership inference and model inversion attacks.

Our solution is a DP mechanism that modifies and normalizes the confidence score vectors to confuse the attacker’s classifier. As such, the only parameter that needs to be tuned is the privacy budget, which controls the amount of perturbation added to the vector. For example, when a learning model is very confident in predicting a given data record, the output confidence score vector will have a very high probability of predicting one class and very low probabilities of predicting the others. By using our method to modify the confidence score vector, the probability distribution can be reshaped so that one class has a slightly higher probability for prediction than other classes. This new confidence score vector appears less confident than the original one and can be used to confuse the attacker’s classifier.

The idea of modifying confidence score vectors to defend against data inference attacks is not new. However, the methods based on this premise, such as those in [53, 106], do not provide a privacy guarantee plus they can only defend against membership inference attacks. By comparison, our method has a privacy guarantee, and offers protection against both membership inference and model inversion attacks. Further, our method preserves the order of scores in confidence score vectors, which guarantees zero classification accuracy loss. Moreover, as our method does not require training any new models, time-efficiency can be achieved. In summary, this chapter makes the following contributions:

- We are the first to propose a one-parameter defense method that requires only one parameter to be tuned, the privacy budget. This method guarantees both DP and time-efficiency against both membership inference and model inversion attacks.
- We theoretically demonstrate how to tune the privacy budget to defend against both types of attacks, while controlling the utility loss of confidence score vectors.
- We empirically show that the presented method effectively mitigates both types of attacks with no loss of classification accuracy, zero training time, and very low test time.

## 6.2 Related work

Defense methods can be roughly classified into four categories based on the adopted techniques: regularization, adversarial examples, deep neural networks and DP.

### 6.2.1 Regularization-based defense methods

Shokri et al. [106] investigated four methods of defense against membership inference attacks. The first method is to restrict the prediction vector to the top  $k$  classes, where a smaller  $k$  means less information is leaked. The second method is to coarsen the precision of the confidence score vector by rounding the classification probabilities within the vector down to  $d$  floating point digits. Again, a smaller  $d$  means less information is leaked. The third method is to increase the entropy of the confidence score vector via a softmax function with a temperature  $t$  to compute the output of the logits vector. The fourth method is to use  $L_2$ -norm standard regularization.

Nasr et al. [79] formalized the interactions between their defense method and a membership inference attack as a min-max privacy game. To find the solution to the game, they train a target model using an adversarial process that minimizes both the prediction loss of the model and the maximum gain of the inference attacks. Through this approach, the target model provides both membership privacy and strong regularization capability.

Salem et al. [98] proposed two defense methods against membership inference attacks. The first method is called 'dropout' which randomly deletes a fixed proportion of edges from a fully connected neural network model in each training iteration to avoid overfitting. The second method is called 'model stacking', which is based on ensemble learning and constructs the target model using three different machine learning models. Two models are placed in the first layer to take the original training data while the third is trained with the confidence score vectors of the first two models. The idea of model stacking is to arrange multiple models in a hierarchy so as to avoid overfitting.

### 6.2.2 Adversarial example-based defense methods

Jia et al. [53] proposed a defense method named MemGuard that adds noise to each confidence score vector to make it an adversarial example. Their idea is based on the fact that deep learning models can be misled by adversarial examples to produce wrong predictions [44, 89]. They formalized the process of adding noise as an optimization problem and developed an algorithm to solve the problem based on gradient descent.

### 6.2.3 Deep neural network-based defense methods

Yang et al. [132] designed a purifier model that takes a confidence score vector as input and reshapes it to meet defense goals. The purifier model consists of an encoder

and a decoder. The encoder maps the confidence score vector predicted by the target model to a latent representation. The decoder then maps the latent representation to a reconstruction of the confidence score vector.

## 6.2.4 Differential privacy-based defense methods

Differential privacy has been a prevalent tool to preserve the privacy of deep learning models [1, 105]. A comprehensive survey regarding DP in deep learning can be found in [43, 51]. To implement differentially private deep learning, noise can be added to one of the five places in a deep neural network: input datasets [47], loss functions [141], gradients [1, 17], weights of neural networks [52, 92], and output classes [87, 90].

Heikkila et al. [47] proposed a general approach for a privacy-preserving learning schema for distributed settings. Their approach combines secure multiparty communication with differentially private Bayesian learning methods. In their approach, each client adds a Gaussian noise to the data and divides the noised data into shares. The shares are not divided independently. Instead, they are divided using a fixed-point representation of real numbers which allows exact cancellation of the noise in the sum. Each share is then sent to a server. This way, the sum of the shares discloses the real value, but separately they are just random noise.

Zhao et al. [141] proposed a privacy-preserving collaborative deep learning system. The system allows users to collaboratively build a collective learning model while only sharing the parameters, not the data. To preserve the private information embodied in the parameters, they developed a functional mechanism, an extended version of the Laplace mechanism, to perturb the objective function of the neural network.

Cheng et al. [17] developed a privacy-preserving algorithm for distributed learning based on a leader-follower framework, where the leaders guide the followers in the right direction to improve their learning speed. For efficiency, communication is limited to leader-follower pairs. To preserve the privacy of the leaders, Gaussian noise is added to the gradients of the leaders' learning models.

Phan et al. [92] proposed a heterogeneous Gaussian mechanism to preserve privacy in deep neural networks. Unlike a regular Gaussian mechanism, this heterogeneous Gaussian mechanism can arbitrarily redistribute noise from the first hidden layer and the gradient of the model to achieve an ideal trade-off between model utility and privacy loss. To obtain the property of arbitrary redistribution, they introduce a noise redistribution vector that can be used to change the variance of the Gaussian distribution. Further, it can be guaranteed that, by adapting the values of the scores in the noise redistribution

vector, more noise can be added to the more vulnerable components of the model to improve robustness and flexibility.

Papernot et al. [87] developed a model called Private Aggregation of Teacher Ensembles (PATE) which has been successfully applied to generative adversarial nets (GANs) for a privacy guarantee [54]. PATE consists an ensemble of  $n$  teacher models; an aggregation mechanism; and a student model. Each teacher model is trained independently on a subset of private data. To protect the privacy of the data labels, Laplace noise is added to the output classes, i.e., the teacher votes. Last, the student model is trained through knowledge transfer from the teacher ensemble with the public data and privacy-preserving labels. Later, Papernot et al. [90] improved the PATE model to make it applicable to large-scale tasks and real-world datasets.

In addition to the utilization of standard DP, local DP has also been adopted recently. For example, Kim et al. [57] adopted Gaussian mechanism to preserve local DP of user data in federated learning models. They also analyzed the trade-offs between user privacy, global utility and transmission rate, where a larger noise variance guarantees a stronger privacy with a lower utility bound and a higher transmission rate bound.

### 6.2.5 Discussion of related work

Among all these defense methods, the regularization and adversarial example methods are only designed to defend against membership inference attacks. The DP methods have mostly been developed for general privacy preservation rather than to protect against a specific inference attack. Only the deep neural network-based method [132] has been specifically developed to defend against both membership inference and model inversion attacks. That method, however, does require new deep neural networks to be trained. Hence, it is not an optimal strategy if time efficiency is a concern.

Our strategy of modifying then normalizing the output confidence score vectors with a DP mechanism provides protection against both membership inference and model inversion attacks, compared to other DP methods which only provide general privacy preservation during the training. Unlike the deep neural network-based method in [132], no new models need to be trained, so our method is very time-efficient. Plus, the order of scores in the confidence vectors is preserved, which skirts the accuracy trade-off inherent to other DP methods.

## 6.3 Preliminaries

### 6.3.1 Data inference attacks

Data inference attacks can come in the form of either a membership inference or a model inversion attack, each of which has a different inference goal. In both of these attacks, the target model is evaluated by its owner but is open to public users, including attackers. The model can be accessed only in a black-box manner, where an attacker inputs a data sample into the model and receives the corresponding output. The real output confidence scores are modified by the owner while only the modified scores are published to the attacker. The modification process is integrated into the model and thus, is hidden to the attacker. Hence, the attacker cannot access any intermediate classification values, and he can only exploit the modified scores to invert the model or infer the membership of samples. Moreover, the structure and parameters of the target model are also unknown to the attacker. The attacker is aware of the distribution of the training dataset of the target model and can collect a new dataset based on the same distribution. The attacker, however, cannot directly query the training dataset of the target model. These assumptions regarding the attacker are the same as those made in [133].

#### 6.3.1.1 Model inversion attacks

Model inversion attacks aim to reconstruct the input data from the confidence score vectors predicted by the target model [37]. The attacker trains a separate attack model on an auxiliary dataset which acts as the inverse of the target model [133]. The attack model takes the confidence score vectors of the target model as input and tries to output the original input data of the target model.

Formally, let  $F$ , again, be the target model and  $G$  be the attack model. Given a data record  $(\mathbf{x}, y)$ , the attacker inputs  $\mathbf{x}$  into  $F$  and receives  $F(\mathbf{x})$ , and then feeds  $F(\mathbf{x})$  into  $G$  and receives  $G(F(\mathbf{x}))$  which is expected to be very similar to  $\mathbf{x}$ , i.e.,  $G(F(\mathbf{x})) \approx \mathbf{x}$ .

In the first phase, the range  $[0, 1)$  is divided into  $k$  non-overlapping sub-ranges. This division is based on the original scores in vector  $\mathbf{y}$  to ensure that each sub-range covers an original score and also, a sub-range with smaller values covers a smaller score. Specifically, suppose the scores in  $\mathbf{y}$  have been ranked as  $y_1 \leq \dots \leq y_k$ . Then, each sub-range  $i$  is  $[\frac{y_{i-1} + y_i}{2}, \frac{y_i + y_{i+1}}{2})$ , where  $2 \leq i \leq k - 1$ . For  $i = 1$ , the sub-range is  $[0, \frac{y_1 + y_2}{2})$ , and for  $i = k$ , the sub-range is  $[\frac{y_{k-1} + y_k}{2}, 1)$ . Then, each sub-range is uniformly discretized to a set containing  $m$  scores. The exponential mechanism is used to select a score  $y'_i$  from



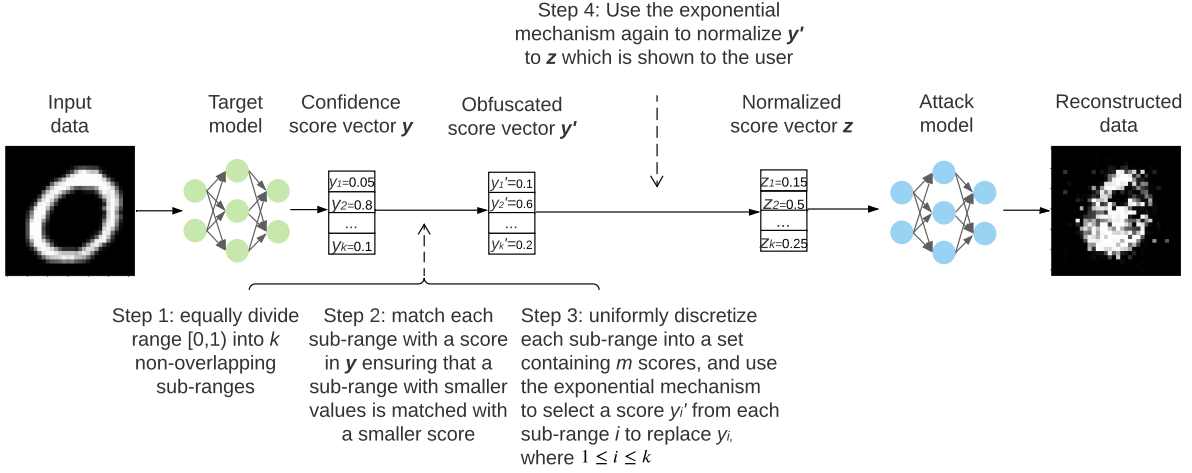


Figure 6.2: Overview of our method. The method consists of four steps. In Step 1, the range  $[0, 1)$  is divided into  $k$  non-overlapping sub-ranges, where  $k$  is the number of classes that the target model  $T$  can classify. In Step 2, each sub-range is matched with a score in vector  $y$ , where  $y$  is output by  $T$ , ensuring that a sub-range with smaller values is matched with a smaller score. In Step 3, each sub-range is uniformly discretized into a set with  $m$  scores, where  $m$  is a hyper-parameter. Then, for each sub-range  $i$ , the exponential mechanism is used to select a score from  $m$  scores. The selected score is named  $y'_i$  used to replace  $y_i$ . In Step 4, since it is likely that  $\sum_{i=1}^k y'_i \neq 1$ , the exponential mechanism is used again to normalize vector  $y'$  to  $z$  which is shown to the attacker.

the  $m$  scores in each sub-range  $i$  to replace  $y_i$ , where  $1 \leq i \leq k$ . In this way, the order of the scores in  $y$  can be preserved in  $y'$  after replacement. As the user of the target model will select the predicted class with the highest score, our method can guarantee zero accuracy loss. In addition, the aim of discretizing each sub-range is to enable the use of an exponential mechanism to select a value. Since the exponential mechanism guarantees DP, the attacker cannot deduce the real value from the selected value. By comparison, other selection methods, e.g., uniformly selection, may not guarantee DP.

In the second phase, since the  $k$  scores in  $y'$  may not sum to 1, we use the exponential mechanism again to normalize them to form a valid confidence score vector  $z$ . There are two reasons to use an exponential mechanism for normalization. First, with exponential normalization, the utility of vectors can be adjusted by choosing different  $\epsilon$  values, which allows us to precisely control the difference in the content of information between a confidence score vector and its normalized version. This control, however, may not be achieved using other normalization methods, e.g., the softmax. The second reason is that by using the exponential normalization, only one parameter  $\epsilon$  needs to be tuned to achieve both the differentially private modifications and the normalization, which

matches the title of the chapter: one parameter defense.

We use an example to explain how the method works and guarantees zero classification accuracy loss of the target model. Suppose we have a confidence score vector with two scores:  $\mathbf{y} = \langle y_1 = 0.2, y_2 = 0.8 \rangle$ . First, we divide the range  $[0, 1)$  into  $k = 2$  non-overlapping sub-ranges based on the scores in  $\mathbf{y}$  as  $[0, \frac{0.2+0.8}{2})$  and  $[\frac{0.2+0.8}{2}, 1)$ . Then, sub-range  $[0, 0.5)$  is matched with  $y_1$ , while sub-range  $[0.5, 1)$  is matched with  $y_2$ , recalling that a sub-range with smaller values is matched with a smaller score. After that, each sub-range is discretized to a set containing  $m$  scores. Suppose  $m$  is set to 5, then, sub-range  $[0, 0.5)$  becomes  $\{0, 0.1, 0.2, 0.3, 0.4\}$  and sub-range  $[0.5, 1)$  becomes  $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ . For each score in  $\mathbf{y}$ , we use the exponential mechanism to select a score in the corresponding discretized sub-range to replace that score. Thus, for  $y_1$ , we use the exponential mechanism to select a score in  $\{0, 0.1, 0.2, 0.3, 0.4\}$  to replace the score of  $y_1$ . Similarly, for  $y_2$ , the replacing score is selected in  $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ . The detail of the selection will be given in Sub-section 6.3.2. Let the scores selected in  $\{0, 0.1, 0.2, 0.3, 0.4\}$  and  $\{0.5, 0.6, 0.7, 0.8, 0.9\}$  be  $y'_1$  and  $y'_2$ , respectively, thus we have  $y'_1 < y'_2$ . Hence, when we use  $y'_1$  and  $y'_2$  to replace  $y_1$  and  $y_2$ , respectively, the order of the scores in  $\mathbf{y}$  can be preserved in  $\mathbf{y}'$  after replacement. This means that the highest score in  $\mathbf{y}$ , after replacement, is still the highest in  $\mathbf{y}'$ . Next, suppose the selected scores are  $y'_1 = 0.4$  and  $y'_2 = 0.9$ . Note that  $0.4 + 0.9 \neq 1$ . Then, we use the exponential mechanism again to normalize  $\mathbf{y}'$ . The aim of the normalization is to ensure that  $\sum_{i=1}^k z_i = 1$ , where each  $z_i$  is computed based on the scores in  $\mathbf{y}'$ . The detail of the normalization will be given in Sub-section 6.3.3. This normalization is a requirement of machine learning classifiers, i.e., presenting normalized vectors to users. The normalization result is, say,  $\mathbf{z} = \langle 0.3, 0.7 \rangle$ , which is given to the attacker. In Section 6.4, we will prove that the normalization can preserve the order of scores in  $\mathbf{y}'$  in the normalized vector  $\mathbf{z}$ . As the predicted class is selected only with the highest score, our method can guarantee zero accuracy loss.

### 6.3.2 Phase 1: Modify the confidence score vector

The first phase of our method is formalized in **Algorithm 12**. In summary, the method takes a confidence score vector  $\mathbf{y}$  as input and outputs a modified vector  $\mathbf{y}'$ . In Line 4, the range  $[0, 1)$  is divided into  $k$  non-overlapping sub-ranges based on  $y_1, \dots, y_k$ . In Lines 5 and 6, each sub-range is uniformly discretized to a set containing  $m$  scores. In Lines 8 and 9, the exponential mechanism is used to randomly select a score  $y'_i$  in each sub-range  $i$  to replace  $y_i$ . As described in Definition 2.8, which score is selected in each sub-range  $i$  is based on its utility. The utility of the  $j^{th}$  score is then set to

**Algorithm 12** Division and modification

- 
- 1: **Input:** A confidence score vector  $\mathbf{y} = \langle y_1, \dots, y_k \rangle$ ;
  - 2: **Output:** A modified confidence score vector  $\mathbf{y}' = \langle y'_1, \dots, y'_k \rangle$ ;
  - 3: Sort  $y_1, \dots, y_k$  such that  $y_1 \leq \dots \leq y_k$ ;
  - 4: Divide range  $[0, 1)$  into  $k$  sub-ranges based on  $y_1, \dots, y_k$ :  $[0, \frac{y_1+y_2}{2})$ ,  $[\frac{y_1+y_2}{2}, \frac{y_2+y_3}{2})$ , ...,  $[\frac{y_{k-1}+y_k}{2}, 1)$ ;
  - 5: **for**  $i = 1$  **to**  $k$  **do**
  - 6:     Uniformly discretize sub-range  $[\frac{y_{i-1}+y_i}{2}, \frac{y_i+y_{i+1}}{2})$  to  $\{\frac{y_{i-1}+y_i}{2}, \frac{y_{i-1}+y_i}{2} + \rho, \dots, \frac{y_{i-1}+y_i}{2} + (m-1)\rho\}$ , where  $\rho = \frac{y_{i+1}-y_{i-1}}{2m}$  and  $m$  is a positive integer representing the granularity of discretization;
  - 7: **end for**
  - 8: **for**  $y_i = y_1$  **to**  $y_k$  **do**
  - 9:     Use the exponential mechanism to randomly select a value from the corresponding discretized sub-range  $i$  as the modified score,  $y'_i$ ;
  - 10: **end for**
- 

$$u_j^i = \frac{1}{|y_i - (\frac{i-1}{k} + (j-1)\rho)|}$$

Scores with a smaller difference to  $y_i$  have a higher utility and thus a higher probability of being selected. Thus, in the sub-range  $i$ , the probability of selecting the  $j^{\text{th}}$  score is proportional to  $\exp(\frac{cu_j^i}{2\Delta u})$ .

In **Algorithm 12**, we need to conduct  $k$  samplings, i.e., selections, and the size of each sampling domain is  $m$ . To avoid a large computation overhead, a finite and small sampling domain is necessary [39]. To limit the size of the sampling domain, we can tune the value of  $m$ . In the experiments, we set  $m = 5$ , i.e., the final result is selected from 5 candidates using the exponential mechanism. The set of  $m = 5$  yields only a small sampling domain and does not introduce a large computation overhead.

### 6.3.3 Phase 2: Normalizing the confidence score vector

The modifications to the confidence score vector  $\mathbf{y}$  in Phase 1 may result in an invalid probability distribution, i.e., where the sum of the scores in  $\mathbf{y}'$  does not equal 1. Hence, Phase 2 involves using the exponential mechanism again to normalize  $\mathbf{y}'$ . This procedure is as follows.

Let the modified vector be  $\mathbf{y}' = \langle y'_1, \dots, y'_k \rangle$ . Then,  $\mathbf{y}'$  can be perturbed into  $\mathbf{z}$ , where each  $z_i$  is computed using the following equation.

$$(6.1) \quad z_i = \frac{\exp(\frac{\epsilon u(\mathbf{y}', y'_i)}{2\Delta u})}{\sum_{1 \leq j \leq k} \exp(\frac{\epsilon u(\mathbf{y}', y'_j)}{2\Delta u})}$$

Theoretically,  $u(\mathbf{y}', y'_i)$  can be set to any value which is positively correlated with  $y'_i$ , i.e., a larger  $y'_i$  should be assigned a larger  $u(\mathbf{y}', y'_i)$ . Specifically, by setting  $u(\mathbf{y}', y'_i) = y'_i$ , i.e., setting the utility of each confidence score to the same value as the score, the sensitivity  $\Delta u$  becomes 1 which is easy for both theoretically analyzing the properties of our defense method and experimentally evaluating its performance. Therefore, Equation 6.1 can be simplified to

$$(6.2) \quad z_i = \frac{\exp(\frac{\epsilon y'_i}{2})}{\sum_{1 \leq j \leq k} \exp(\frac{\epsilon y'_j}{2})}$$

The resulting vector  $\mathbf{z}$  is the final output.

### 6.3.4 Discussion of the method

In our method, the privacy protection is on confidence score vectors  $\mathbf{y}$  rather than the original training dataset  $D_{\text{target}}^{\text{train}}$ . This is because 1) attackers in our problem are not allowed to directly access the training dataset, and 2) model inversion attacks are not against the training dataset, instead, these attacks aim to reconstruct any input data to the target model. The attackers can access the target model and exploit the prediction results of the target model to launch both membership inference and model inversion attacks. Therefore, our protection focuses on the prediction results of the target model, i.e., confidence score vectors.

Although our method is applied only to the prediction results of the target model, it can still defend against both membership inference and model inversion attacks. The attacker queries the target model  $F$  by feeding a data record  $\mathbf{x}$  to it, and expects to receive a response  $F(\mathbf{x})$ . However, by using our private prediction interface  $\mathcal{M}_d$ , the response  $F(\mathbf{x})$  is obfuscated to  $\mathcal{M}_d(F(\mathbf{x}))$ . Since the existing attack methods must use  $F(\mathbf{x})$  to launch the membership inference and model inversion attacks, altering  $F(\mathbf{x})$  to  $\mathcal{M}_d(F(\mathbf{x}))$  can significantly reduce the attack precision. This can be further explained by the fact that the attack model  $G$  is trained using  $F(\mathbf{x})$  as input, i.e.,  $G(F(\mathbf{x}))$ , where for membership inference attacks,  $G(F(\mathbf{x})) = Pr(\mathbf{x} \in D_{\text{target}}^{\text{train}})$ , and for model inversion attacks,  $G(F(\mathbf{x})) = \hat{x}$ . Now,  $F(\mathbf{x})$  is obfuscated to  $\mathcal{M}_d(F(\mathbf{x}))$  which is used as input to the attack

model  $G$ , i.e.,  $G(\mathcal{M}_d(F(\mathbf{x})))$ . To achieve a precise attack, the attacker has to guarantee that  $G(\mathcal{M}_d(F(\mathbf{x}))) = G(F(\mathbf{x}))$ , which means that  $\mathcal{M}_d(F(\mathbf{x}))$  and  $F(\mathbf{x})$  must be in the same class. However, there is no guarantee that an input can be in the same class as its obfuscated version with differentially private noise. Moreover, as  $\mathcal{M}_d$  satisfies DP, the attacker cannot deduce  $F(\mathbf{x})$  from  $\mathcal{M}_d(F(\mathbf{x}))$ . In the experiments, we have also attempted to train the attack model  $G$  using  $\mathcal{M}_d(F(\mathbf{x}))$  as input, but this training did not converge.

## 6.4 Properties of the defense method and how to use them to defend against attacks

This section begins with the proof that our defense method satisfies DP. Also, to maintain the DP guarantee, a bound is set to limit the number of queries that an attacker can access the target model  $F$  using the same input data record  $\mathbf{x}$  (Sub-section 6.4.1). Then, an analysis of tuning  $\epsilon$  follows, which details the various properties of our method with different  $\epsilon$  values (Sub-section 6.4.2). After that, we explain how to use these properties to defend against the data inference attacks (Sub-sections 6.4.3 and 6.4.4).

### 6.4.1 Privacy analysis

**Lemma 6.1** (Post-processing theorem [30]). *Let  $\mathcal{M}_1 : Y \rightarrow Y'$  be a randomized algorithm that is  $\epsilon$ -differentially private. Let  $\mathcal{M}_2 : Y' \rightarrow Z$  be any mapping including deterministic functions. Then  $\mathcal{M}_1 \circ \mathcal{M}_2 : Y \rightarrow Z$  is  $\epsilon$ -differentially private.*

Lemma 6.1 states that the combination of a differentially private algorithm and a deterministic algorithm still guarantees DP.

**Theorem 6.1.** *Algorithm 12 satisfies  $(k \cdot \epsilon)$ -DP, where  $\epsilon$  is the privacy budget and  $k$  is the number of scores in a confidence score vector  $\mathbf{y}$ , i.e., the number of classes that the target model  $F$  can classify.*

**Proof.** In Line 8 of **Algorithm 12** (the first phase of our method), the exponential mechanism is used  $k$  times to randomly select  $k$  scores to replace  $y_1, \dots, y_k$ , i.e., to perform the mapping from  $\mathbf{y}$  to  $\mathbf{y}'$ . According to Definition 2.8, exponential mechanism defines a utility function  $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \rightarrow \mathbb{R}$ , which maps dataset-output pairs to utility scores. In our problem, for each sub-range  $i$ :  $[\frac{y_{i-1}+y_i}{2}, \frac{y_i+y_{i+1}}{2})$ , when  $2 \leq i \leq k-1$ , and  $[0, \frac{y_1+y_2}{2})$  and  $[\frac{y_{k-1}+y_k}{2}, 1)$  when  $i=1$  and  $i=k$ , respectively, by applying Definition 2.8,

we interpret dataset  $D$  as a confidence score vector  $\mathbf{y}$ , and interpret output set  $\mathcal{R}$  as  $\mathcal{R} = \{\frac{y_{i-1}+y_i}{2}, \frac{y_{i-1}+y_i}{2} + \rho, \dots, \frac{y_{i-1}+y_i}{2} + (m-1)\rho\}$ . In addition, we interpret a neighboring dataset  $D'$  as another confidence score vector  $\hat{\mathbf{y}}$ , where  $\|\hat{\mathbf{y}} - \mathbf{y}\|_1 \leq 1$  and  $y_i + y_{i+1} = \hat{y}_i + \hat{y}_{i+1}$ , where  $1 \leq i \leq k-1$ . Then,  $\mathbf{y}$  and  $\hat{\mathbf{y}}$  have the same output set  $\mathcal{R}$ .

Let the probability of selecting the  $j^{\text{th}}$  score  $r_j$  from  $\mathcal{R}$  with vector  $\mathbf{y}$  be  $P(r_j|\mathbf{y})$  and the probability of selecting  $r_j$  from  $\mathcal{R}$  with a neighboring vector  $\hat{\mathbf{y}}$  be  $P(r_j|\hat{\mathbf{y}})$ . Also, let  $u^i(\mathbf{y}, r_j)$  and  $u^i(\hat{\mathbf{y}}, r_j)$  be the utility of selecting  $r_j$  from  $\mathcal{R}$  with  $\mathbf{y}$  and  $\hat{\mathbf{y}}$ , respectively.

Then, we have  $\frac{P(r_j|\mathbf{y})}{P(r_j|\hat{\mathbf{y}})} = \left[ \frac{\exp(\frac{eu^i(\mathbf{y}, r_j)}{2\Delta u})}{\sum_{r_{j'} \in \mathcal{R}} \exp(\frac{eu^i(\mathbf{y}, r_{j'})}{2\Delta u})} \right] \bigg/ \left[ \frac{\exp(\frac{eu^i(\hat{\mathbf{y}}, r_j)}{2\Delta u})}{\sum_{r_{j'} \in \mathcal{R}} \exp(\frac{eu^i(\hat{\mathbf{y}}, r_{j'})}{2\Delta u})} \right]$ . Based on the knowledge of exponential mechanism [30], we have  $\frac{P(r_j|\mathbf{y})}{P(r_j|\hat{\mathbf{y}})} \leq \exp(\epsilon)$ .

By symmetry,  $\frac{P(r_j|\mathbf{y})}{P(r_j|\hat{\mathbf{y}})} \geq \exp(-\epsilon)$ . Therefore, in each sub-range  $i$ , our method satisfies  $\epsilon$ -DP. Since there are  $k$  sub-ranges and the exponential mechanism is used in each sub-range, **Algorithm 12** gives  $(k \cdot \epsilon)$ -DP.

The second phase of our method is deterministic, mapping a differentially private vector  $\mathbf{y}'$  to  $\mathbf{z}$ . Therefore, according to Lemma 6.1, the combination of the first and second phases still guarantee  $(k \cdot \epsilon)$ -DP. ■

**Theorem 6.2.** *The defense method,  $\mathcal{M}_d$ , is a  $(k \cdot \epsilon)$ -differentially private prediction interface.*

**Proof.** Theorem 6.1 shows that for each input query  $\mathbf{x}$  of the target model  $F$ , the defense method  $\mathcal{M}_d$  can obfuscate the corresponding response, i.e., output vector  $\mathbf{y}$ , in a differentially private manner. According to Definition 2.4, as the sequence of queries and responses satisfies  $(k \cdot \epsilon)$ -DP, the defense method  $\mathcal{M}_d$  is a  $(k \cdot \epsilon)$ -differentially private prediction interface. ■

Theorems 6.1 and 6.2 and the accompanying proof demonstrate that our defense method  $\mathcal{M}_d$  provides a privacy guarantee. By giving DP guarantee, an attacker cannot deduce the original vector  $\mathbf{y}$  from the perturbed vector  $\mathbf{z}$ . As shown in the experiments, based on  $\mathbf{z}$ , the attacker can neither precisely reconstruct the input data record  $\mathbf{x}$  nor successfully infer whether  $\mathbf{x}$  is in the training set of the target model  $F$ . However, if an attacker uses the same input data record  $\mathbf{x}$  to access the target model  $F$  multiple times, the attacker may deduce the original vector  $\mathbf{y}$  by observing the perturbed vectors. This is because in DP, a privacy budget is used to control the privacy level. Every time an original vector is perturbed and released, the privacy budget is partially consumed. Once the privacy budget is used up, DP cannot guarantee the privacy of the original vector

anymore. To guarantee the privacy level of an original vector, a bound must be set on the number of times that a user can access the target model using the same data record. The detailed computation of the bound is as follows.

**Definition 6.1** (KL-Divergence [30]). The KL-Divergence between two random variables  $Y$  and  $Z$  taking values from the same domain is defined to be:

$$(6.3) \quad D(Y||Z) = \mathbb{E}_{y \sim Y} \left[ \ln \frac{\Pr(Y = y)}{\Pr(Z = y)} \right].$$

**Definition 6.2** (Max Divergence [30]). The Max Divergence between two random variables  $Y$  and  $Z$  taking values from the same domain is defined to be:

$$(6.4) \quad D_\infty(Y||Z) = \max_{S \subseteq \text{Supp}(Y)} \left[ \ln \frac{\Pr(Y \in S)}{\Pr(Z \in S)} \right].$$

**Lemma 6.2** ([30]). A mechanism  $\mathcal{M}$  is  $\epsilon$ -differentially private if and only if on every two neighboring datasets  $x$  and  $x'$ ,  $D_\infty(\mathcal{M}(x)||\mathcal{M}(x')) \leq \epsilon$  and  $D_\infty(\mathcal{M}(x')||\mathcal{M}(x)) \leq \epsilon$ .

**Lemma 6.3** ([30]). Suppose that random variables  $Y$  and  $Z$  satisfy  $D_\infty(Y||Z) \leq \epsilon$  and  $D_\infty(Z||Y) \leq \epsilon$ . Then,  $D(Y||Z) \leq \epsilon \cdot (e^\epsilon - 1)$ .

**Theorem 6.3.** Given that the privacy level of an original vector is  $k \cdot \epsilon$  for each access, to guarantee its overall privacy level to be  $\epsilon'$ , the upper bound of the number of rounds is  $\frac{\epsilon' \cdot (e^{\epsilon'} - 1)}{k \cdot \epsilon \cdot (e^{k\epsilon} - 1)}$ .

**Proof.** Let the upper bound of the number of access times be  $b$ , and the corresponding  $b$  perturbed vectors, which can be observed by an attacker, be  $\mathbf{o} = (\mathbf{z}^1, \dots, \mathbf{z}^b)$ . We have

$$\begin{aligned} D(Y||Z) &= \ln \left[ \frac{\Pr(Y = \mathbf{o})}{\Pr(Z = \mathbf{o})} \right] = \ln \left[ \prod_{i=1}^b \frac{\Pr(Y_i = \mathbf{z}^i)}{\Pr(Z_i = \mathbf{z}^i)} \right] \\ &= \sum_{i=1}^b \ln \left[ \frac{\Pr(Y_i = \mathbf{z}^i)}{\Pr(Z_i = \mathbf{z}^i)} \right] = \sum_{i=1}^b D(Y_i||Z_i). \end{aligned}$$

As the original vector is guaranteed  $k \cdot \epsilon$ -DP for each individual access, based on Lemma 6.2, we have  $D_\infty(Y_i||Z_i) \leq k \cdot \epsilon$  and  $D_\infty(Z_i||Y_i) \leq k \cdot \epsilon$ . Based on this result, according to Lemma 6.3, we have  $D(Y_i||Z_i) \leq k \cdot \epsilon \cdot (e^{k\epsilon} - 1)$ . Thus, we have  $D(Y||Z) = \sum_{i=1}^b D(Y_i||Z_i) \leq b \cdot k \cdot \epsilon \cdot (e^{k\epsilon} - 1)$ .

To guarantee the original vector's overall privacy level to be  $\epsilon'$ , according to Lemma 6.2, we have  $D_\infty(Y||Z) \leq \epsilon'$  and  $D_\infty(Z||Y) \leq \epsilon'$ . By using Lemma 6.3, we have  $D(Y||Z) \leq \epsilon' \cdot (e^{\epsilon'} - 1)$ . As  $D(Y||Z) \leq b \cdot k \cdot \epsilon \cdot (e^{k\epsilon} - 1)$  and  $D(Y||Z) \leq \epsilon' \cdot (e^{\epsilon'} - 1)$ , the upper bound  $b$  is limited by  $\frac{\epsilon' \cdot (e^{\epsilon'} - 1)}{k \cdot \epsilon \cdot (e^{k\epsilon} - 1)}$ . ■

## 6.4.2 Analysis of tuning $\epsilon$

The input for the first step of our method is a confidence score vector  $\mathbf{y}$ , and the output is a perturbed vector  $\mathbf{y}'$ . The input for the second step is  $\mathbf{y}'$ , and the output is a normalized vector  $\mathbf{z}$ . To maximize the utility of the normalized vector, it should be equal to the original vector:  $\mathbf{z} = \mathbf{y}$ . This equality does not guarantee any privacy over the original vector and must be avoided in practice, but this equality gives us a start point to investigate how to tune the parameter,  $\epsilon$ , to achieve the balance between privacy and utility. Specifically, the utility of a normalized vector  $\mathbf{z}$  is defined as  $\|\mathbf{z} - \mathbf{y}\|_1$ , where a smaller value of  $\|\mathbf{z} - \mathbf{y}\|_1$  means a higher utility but a lower privacy guarantee.

Given that  $\mathbf{y} = \langle y_1, \dots, y_k \rangle$ ,  $\mathbf{y}' = \langle y'_1, \dots, y'_k \rangle$  and  $\mathbf{z} = \langle z_1, \dots, z_k \rangle$ , we have the following system of  $k$  equations:

$$(6.5) \quad \begin{cases} z_1 = \frac{\exp(\frac{\epsilon y'_1}{2})}{\sum_{1 \leq j \leq k} \exp(\frac{\epsilon y'_j}{2})} = y_1, \\ \dots \\ z_k = \frac{\exp(\frac{\epsilon y'_k}{2})}{\sum_{1 \leq j \leq k} \exp(\frac{\epsilon y'_j}{2})} = y_k. \end{cases}$$

We first prove that **Algorithm 12** preserves the order of scores in vector  $\mathbf{y}$ , and then show that Equation 6.5 has a unique positive solution. After that, we analyze the properties of the solution to Equation 6.5.

**Lemma 6.4.** *Algorithm 12 preserves the order of scores in vector  $\mathbf{y}$ , i.e., if  $y_i < y_j$ , then  $y'_i < y'_j$ , where  $1 \leq i \neq j \leq k$ .*

**Proof.** According to Line 3 of **Algorithm 12**, when  $y_i < y_j$ , we have  $i < j$ . According to Line 8 of **Algorithm 12**, we have  $y'_i \in [\frac{y_{i-1} + y_i}{2}, \frac{y_i + y_{i+1}}{2})$  and  $y'_j \in [\frac{y_{j-1} + y_j}{2}, \frac{y_j + y_{j+1}}{2})$ . As  $i < j$ , we have  $i \leq j - 1$  and  $i + 1 \leq j$ . Thus, we have  $y_i \leq y_{j-1}$  and  $y_{i+1} \leq y_j$ , which implies that  $\frac{y_i + y_{i+1}}{2} \leq \frac{y_{j-1} + y_j}{2}$ . Therefore, we can conclude  $y'_i < y'_j$ . ■

The following Lemmas 6.5 and 6.6 and Theorem 6.4 show that Equation 6.5 has a unique positive solution.

**Lemma 6.5.** *Let  $y'_{min} = \min\{y'_1, \dots, y'_k\}$  and  $y'_{max} = \max\{y'_1, \dots, y'_k\}$ , we have:  $y'_{min} < \frac{1}{k}$  and  $y'_{max} \geq \frac{1}{k}$ .*

**Proof.** The following proof covers  $y'_{max} \geq \frac{1}{k}$ . The proof of  $y'_{min} < \frac{1}{k}$  is similar. According to Lines 4-6 of **Algorithm 12**, we know that  $\frac{y_{k-1} + y_k}{2} \leq y'_{max} < 1$ .



6.4. PROPERTIES OF THE DEFENSE METHOD AND HOW TO USE THEM TO DEFEND AGAINST ATTACKS

---

Thus, to prove  $y'_{max} \geq \frac{1}{k}$ , we need only to prove  $\frac{y_{k-1}+y_k}{2} \geq \frac{1}{k}$ . For this, we use mathematical induction. When  $k = 2$ , we have  $\frac{y_{k-1}+y_k}{2} = \frac{y_1+y_2}{2} = \frac{1}{2} = \frac{1}{k}$ . Thus, the conclusion is established. Assume that when  $k = n$ , the conclusion is also established, i.e.,  $\frac{y_{n-1}+y_n}{2} \geq \frac{1}{n}$ . Next, we prove that when  $k = n + 1$ , the conclusion is still established, i.e., proving  $\frac{y_n+y_{n+1}}{2} \geq \frac{1}{n+1}$ . As  $\frac{y_{n-1}+y_n}{2} \geq \frac{1}{n}$ , we have  $y_n \geq \frac{2}{n} - y_{n-1}$ . Therefore, we have

$$\begin{aligned} \frac{y_n + y_{n+1}}{2} &\geq \frac{(\frac{2}{n} - y_{n-1}) + y_{n+1}}{2} \\ &= \frac{1}{n} + \frac{y_{n+1} - y_{n-1}}{2} \\ &\geq \frac{1}{n} > \frac{1}{n+1}. \end{aligned}$$

The second inequality is based on Line 3 of **Algorithm 12**, where the scores of vector  $\mathbf{y}$  are sorted as  $y_1 \leq \dots \leq y_k$ . Hence,  $y_{n+1} \geq y_{n-1}$  and  $\frac{y_{n+1}-y_{n-1}}{2} \geq 0$ . The lemma has been proven. ■

**Lemma 6.6.** *The system in Equation 6.5 has at least one positive solution.*

**Proof.** According to Equation 6.5, we have  $\frac{y_{max}}{y_j} = \frac{\exp(\frac{\epsilon y'_{max}}{2})}{\exp(\frac{\epsilon y'_j}{2})}$ , where  $1 \leq j \leq k$ . Then, we have the following deduction.

$$\begin{aligned} \frac{y_{max}}{y_j} &= \frac{\exp(\frac{\epsilon y'_{max}}{2})}{\exp(\frac{\epsilon y'_j}{2})} \\ \ln(\frac{y_{max}}{y_j}) &= \ln(\frac{\exp(\frac{\epsilon y'_{max}}{2})}{\exp(\frac{\epsilon y'_j}{2})}) \\ \ln(y_{max}) - \ln(y_j) &= \frac{\epsilon}{2}(y'_{max} - y'_j) \\ \epsilon &= \frac{2[\ln(y_{max}) - \ln(y_j)]}{y'_{max} - y'_j} > 0 \end{aligned}$$

This  $\epsilon$  is a positive solution to the system. ■

Next, we prove that this  $\epsilon$  is a unique solution. Formally, based on Lemmas 6.5 and 6.6, we have:

**Theorem 6.4.** *The system in Equation 6.5 has a unique positive solution.*

**Proof.** Assume we have two different positive solutions  $\epsilon_1$  and  $\epsilon_2$ . Then, we have

$$\begin{aligned} \frac{y_{max}}{y_{min}} &= \frac{\exp(\frac{\epsilon_1 y'_{max}}{2})}{\exp(\frac{\epsilon_1 y'_{min}}{2})} = \frac{\exp(\frac{\epsilon_2 y'_{max}}{2})}{\exp(\frac{\epsilon_2 y'_{min}}{2})} \\ &\Rightarrow \exp[\frac{\epsilon_1 - \epsilon_2}{2} y'_{max}] = \exp[\frac{\epsilon_1 - \epsilon_2}{2} y'_{min}] \\ &\Rightarrow \exp[\frac{\epsilon_1 - \epsilon_2}{2} (y'_{max} - y'_{min})] = 1. \end{aligned}$$

According to Lemma 6.5,  $y'_{max} \neq y'_{min}$ , thus  $\epsilon_1 = \epsilon_2$ . According to Lemma 6.6, the system has at least one solution. Hence, the system has a unique solution.  $\blacksquare$

The properties of the solution are analyzed as follows. Given  $z_i = y_i = \frac{\exp(\frac{\epsilon y'_i}{2})}{\sum_{1 \leq j \leq k} \exp(\frac{\epsilon y'_j}{2})}$ , where  $1 \leq i \leq k$ , the solution has the following properties.

**Property 6.1.** When  $\epsilon > 0$ , if  $y'_i > y'_j$ , then  $z_i > z_j$ , where  $1 \leq i, j \leq k$ .

**Proof.**  $\frac{z_i}{z_j} = \exp[\frac{\epsilon}{2}(y'_i - y'_j)] \geq 1$ , therefore,  $z_i > z_j$ .  $\blacksquare$

Property 6.1 combined with Lemma 6.4 contends that our defense method preserves the order of scores in confidence score vector  $\mathbf{y}$ . As such, it holds that since the order of scores in  $\mathbf{y}$  can be preserved, when  $\epsilon > 0$ , the utility of the confidence score vector  $\mathbf{y}$  can be guaranteed. This means that the class with the highest probability in  $\mathbf{y}$  still has the highest probability in the normalized vector  $\mathbf{z}$ . This property guarantees a good user experience, as users usually select the predicted class with the highest probability.

Property 6.1 combined with Lemma 6.4 contends that our defense method preserves the order of scores in confidence score vector  $\mathbf{y}$ . As such, it holds that since the order of scores in  $\mathbf{y}$  can be preserved, when  $\epsilon > 0$ , the utility of the confidence score vector  $\mathbf{y}$  can be guaranteed. This means that the class with the highest probability in  $\mathbf{y}$  still has the highest probability in the normalized vector  $\mathbf{z}$ . This property guarantees a good user experience, as users usually select the predicted class with the highest probability.

Further discussions on Property 6.1 are included with the Properties 6.2 and 6.3.

**Property 6.2.** Let  $\epsilon^*$  be the solution of the system and  $y'_i > y'_j$ , where  $1 \leq i, j \leq k$ . Then, if  $\epsilon > \epsilon^*$ ,  $\frac{z_i}{z_j} > \frac{y_i}{y_j}$ ; if  $\epsilon < \epsilon^*$ ,  $\frac{z_i}{z_j} < \frac{y_i}{y_j}$ .

**Proof.** Because  $z_i = \frac{\exp(\frac{\epsilon y'_i}{2})}{\sum_{1 \leq j \leq k} \exp(\frac{\epsilon y'_j}{2})}$  and  $y_i = \frac{\exp(\frac{\epsilon^* y'_i}{2})}{\sum_{1 \leq j \leq k} \exp(\frac{\epsilon^* y'_j}{2})}$ , we have

6.4. PROPERTIES OF THE DEFENSE METHOD AND HOW TO USE THEM TO  
DEFEND AGAINST ATTACKS

---

$$\begin{aligned}\frac{z_i/y_i}{z_j/y_j} &= \frac{\exp[\frac{\epsilon}{2}(y'_i - y'_j)]}{\exp[\frac{\epsilon^*}{2}(y'_i - y'_j)]} \\ &= \exp[\frac{\epsilon - \epsilon^*}{2}(y'_i - y'_j)].\end{aligned}$$

Since it is assumed that  $y'_i > y'_j$ , when  $\epsilon > \epsilon^*$ ,  $\frac{z_i/y_i}{z_j/y_j} > 1 \Rightarrow \frac{z_i}{z_j} > \frac{y_i}{y_j}$ ; when  $\epsilon < \epsilon^*$ ,  $\frac{z_i/y_i}{z_j/y_j} < 1 \Rightarrow \frac{z_i}{z_j} < \frac{y_i}{y_j}$ . ■

**Property 6.3.** *If  $\epsilon > \epsilon^*$ , then  $z_{min} < y_{min}$  and  $z_{max} > y_{max}$ ; if  $\epsilon < \epsilon^*$ , then  $z_{min} > y_{min}$  and  $z_{max} < y_{max}$ .*

**Proof.** We know that

$$\begin{aligned}&\sum_{1 \leq i \leq k} \exp[\frac{\epsilon}{2}(y'_i - y'_j)] - \sum_{1 \leq i \leq k} \exp[\frac{\epsilon^*}{2}(y'_i - y'_j)] \\ &= \sum_{1 \leq i \leq k} [\exp[\frac{\epsilon}{2}(y'_i - y'_j)] - \exp[\frac{\epsilon^*}{2}(y'_i - y'_j)]],\end{aligned}$$

where  $1 \leq j \leq k$ . Therefore, when  $\epsilon > \epsilon^*$ , we have

$$\begin{aligned}&\sum_{1 \leq i \leq k} \exp[\frac{\epsilon}{2}(y'_i - y'_{min})] - \sum_{1 \leq i \leq k} \exp[\frac{\epsilon^*}{2}(y'_i - y'_{min})] > 0, \\ &\sum_{1 \leq i \leq k} \exp[\frac{\epsilon}{2}(y'_i - y'_{max})] - \sum_{1 \leq i \leq k} \exp[\frac{\epsilon^*}{2}(y'_i - y'_{max})] < 0.\end{aligned}$$

Because

$$\begin{aligned}\frac{1}{z_{min}} &= \sum_{1 \leq i \leq k} \frac{z_i}{z_{min}} = \sum_{1 \leq i \leq k} \exp[\frac{\epsilon}{2}(y'_i - y'_{min})], \\ \frac{1}{y_{min}} &= \sum_{1 \leq i \leq k} \frac{y_i}{y_{min}} = \sum_{1 \leq i \leq k} \exp[\frac{\epsilon^*}{2}(y'_i - y'_{min})], \\ \frac{1}{z_{max}} &= \sum_{1 \leq i \leq k} \frac{z_i}{z_{max}} = \sum_{1 \leq i \leq k} \exp[\frac{\epsilon}{2}(y'_i - y'_{max})], \\ \frac{1}{y_{max}} &= \sum_{1 \leq i \leq k} \frac{y_i}{y_{max}} = \sum_{1 \leq i \leq k} \exp[\frac{\epsilon^*}{2}(y'_i - y'_{max})],\end{aligned}$$

thus, we have

$$\begin{aligned}\frac{1}{z_{min}} - \frac{1}{y_{min}} &> 0 \Rightarrow z_{min} < y_{min}, \\ \frac{1}{z_{max}} - \frac{1}{y_{max}} &< 0 \Rightarrow z_{max} > y_{max}.\end{aligned}$$

By symmetry, we have that when  $\epsilon < \epsilon^*$ ,  $z_{min} > y_{min}$  and  $z_{max} < y_{max}$ . ■

Properties 6.2 and 6.3 state that if  $\epsilon > \epsilon^*$ , the difference in probabilities  $y_{max} - y_{min}$  within the confidence score vector  $\mathbf{y}$  will increase in vector  $\mathbf{z}$ , i.e.,  $z_{max} - z_{min} > y_{max} - y_{min}$ . This means that even if the target model is not very confident in predicting an input data record  $\mathbf{x}$ , the defense method can make the output appear very confident to the attacker. Similarly, the defense method can also make a very confident output appear less confident to the attacker.

The last property to analyze is the change in the distance between a confidence score vector  $\mathbf{y}$  and its perturbed version  $\mathbf{z}$ . As the success of a model inversion attack is based on the rich information contained in each confidence score vector, an intuitive way to defend against model inversion attacks is to widen the distance between  $\mathbf{y}$  and  $\mathbf{z}$ . The distance is formally defined as  $\|\mathbf{z} - \mathbf{y}\|_1 = \sum_{i=1}^k |z_i - y_i|$ . Then, we have the following property.

**Property 6.4.** *If  $\epsilon > \epsilon^*$ , then  $\|\mathbf{z} - \mathbf{y}\|_1$  increases as  $\epsilon$  increases; if  $\epsilon < \epsilon^*$ , then  $\|\mathbf{z} - \mathbf{y}\|_1$  increases as  $\epsilon$  decreases.*

**Proof.**

$$\begin{aligned} \|\mathbf{z} - \mathbf{y}\|_1 &= \sum_{i=1}^k |z_i - y_i| \\ &= \sum_{i=1}^k \left| \frac{\exp(\frac{\epsilon y'_i}{2})}{\sum_{j=1}^k \exp(\frac{\epsilon y'_j}{2})} - \frac{\exp(\frac{\epsilon^* y'_i}{2})}{\sum_{j=1}^k \exp(\frac{\epsilon^* y'_j}{2})} \right| \\ &= \frac{\sum_{i=1}^k \left| \exp(\frac{\epsilon y'_i}{2}) \sum_{j=1}^k \exp(\frac{\epsilon^* y'_j}{2}) - \exp(\frac{\epsilon^* y'_i}{2}) \sum_{j=1}^k \exp(\frac{\epsilon y'_j}{2}) \right|}{\sum_{j=1}^k \exp(\frac{\epsilon y'_j}{2}) \cdot \sum_{j=1}^k \exp(\frac{\epsilon^* y'_j}{2})} \end{aligned}$$

As  $\sum_{j=1}^k \exp(\frac{\epsilon y'_j}{2}) \cdot \sum_{j=1}^k \exp(\frac{\epsilon^* y'_j}{2}) > 0$ , we focus only on  $\sum_{i=1}^k \left| \exp(\frac{\epsilon y'_i}{2}) \sum_{j=1}^k \exp(\frac{\epsilon^* y'_j}{2}) - \exp(\frac{\epsilon^* y'_i}{2}) \sum_{j=1}^k \exp(\frac{\epsilon y'_j}{2}) \right|$ . Because

$$\begin{aligned} &\sum_{i=1}^k \left| \exp(\frac{\epsilon y'_i}{2}) \sum_{j=1}^k \exp(\frac{\epsilon^* y'_j}{2}) - \exp(\frac{\epsilon^* y'_i}{2}) \sum_{j=1}^k \exp(\frac{\epsilon y'_j}{2}) \right| \\ &= \sum_{i=1}^k \left| \sum_{j=1}^k \exp[\frac{1}{2}(\epsilon y'_i + \epsilon^* y'_j)] - \sum_{j=1}^k \exp[\frac{1}{2}(\epsilon^* y'_i + \epsilon y'_j)] \right|, \end{aligned}$$

thus, evaluating  $\sum_{i=1}^k \left| \sum_{j=1}^k \exp[\frac{1}{2}(\epsilon y'_i + \epsilon^* y'_j)] - \sum_{j=1}^k \exp[\frac{1}{2}(\epsilon^* y'_i + \epsilon y'_j)] \right|$  is equivalent to evaluating  $\sum_{i=1}^k \left| \sum_{j=1}^k (\epsilon y'_i + \epsilon^* y'_j) - \sum_{j=1}^k (\epsilon^* y'_i + \epsilon y'_j) \right|$ . We have

$$\begin{aligned}
& \sum_{i=1}^k \left| \sum_{j=1}^k (\epsilon y'_i + \epsilon^* y'_j) - \sum_{j=1}^k (\epsilon^* y'_i + \epsilon y'_j) \right| \\
&= \sum_{i=1}^k \left| k\epsilon y'_i + \sum_{j=1}^k \epsilon^* y'_j - (k\epsilon^* y'_i + \sum_{j=1}^k \epsilon y'_j) \right| \\
&= \sum_{i=1}^k \left| k y'_i (\epsilon - \epsilon^*) - (\epsilon - \epsilon^*) \sum_{j=1}^k y'_j \right| \\
&= \sum_{i=1}^k |(\epsilon - \epsilon^*) (k y'_i - \sum_{j=1}^k y'_j)| \\
&= |\epsilon - \epsilon^*| \sum_{i=1}^k \left| k y'_i - \sum_{j=1}^k y'_j \right|.
\end{aligned}$$

Hence,  $|\epsilon - \epsilon^*|$  determines the distance between  $\mathbf{y}$  and  $\mathbf{z}$ , and the conclusion of this property is achieved. ■

### 6.4.3 Defending against membership inference attacks

In membership inference attacks, the attacker essentially exploits any overfitting of the target model, in that models often behave more confidently toward data on which they were trained versus data they are seeing for the first time [106]. Thus, the overarching aim of our defense method is to reduce the gap between the confidence score vectors of training set members versus non-members.

To this end, we set a threshold  $\tau$  to  $0 < \tau < 1$ , which is used to decide whether the target model is confident in an input data record. To explain, on the one hand, if  $y_{max} > \tau$ , the target model will appear confident in the input data record. Therefore, according to Property 6.3, the defense method should reduce this confidence to confuse the attacker by setting  $\epsilon < \epsilon^*$ , where  $\epsilon^*$  makes  $\mathbf{z} = \mathbf{y}$ . The exact value of  $\epsilon$  depends on the expected utility of the perturbed confidence score vector:  $\|\mathbf{z} - \mathbf{y}\|_1$ . On the other hand, if  $y_{max} \leq \tau$ , the target model will appear less confident in the input data record. Therefore, the defense method should increase the confidence by setting  $\epsilon > \epsilon^*$ .

Moreover, according to Property 6.1, as the value of  $\epsilon$  is always larger than 0, the order of the scores in  $\mathbf{y}$  will be preserved in  $\mathbf{z}$  after perturbation, i.e., if  $y_i$  is  $y_{max}$  in  $\mathbf{y}$ , then  $z_i$  is  $z_{max}$  in  $\mathbf{z}$ . Typically, the user of the target model will select the predicted class with the highest probability. Hence, this defense method does not affect user experience.

## 6.4.4 Defending against model inversion attacks

In model inversion attacks, the attacker trains an inversion model  $G$  to approximate the inverse mapping of the target model  $F$  [133]. The attack works due to the rich information contained in confidence vectors [37]. Thus, this defense method works to reduce the content of that information. One way to do this is to increase the distance between the confidence vector and its perturbed version. According to Property 6.4, the distance between  $\mathbf{y}$  and  $\mathbf{z}$  is based on  $|\epsilon - \epsilon^*|$ . Hence, to widen the distance, if  $\epsilon > \epsilon^*$ , then  $\epsilon$  should be set a large value. Otherwise,  $\epsilon$  should be set a small value. In particular, when  $\epsilon > \epsilon^*$ , increasing the value of  $\epsilon$  will increase the variance of the scores in  $\mathbf{z}$ . A high variance makes  $\mathbf{z}$  appear as a confident prediction. Oppositely, when  $\epsilon < \epsilon^*$ , increasing the value of  $\epsilon$  will decrease the variance of the scores in  $\mathbf{z}$ . A low variance makes  $\mathbf{z}$  appear as an unconfident prediction. Thus, using a large or small value of  $\epsilon$  depends on the original vector  $\mathbf{y}$ . If  $\mathbf{y}$  is a confident prediction,  $\epsilon$  should be set a small value which not only increases the distance between  $\mathbf{y}$  and  $\mathbf{z}$  but also makes  $\mathbf{z}$  appear as an unconfident prediction. Otherwise, if  $\mathbf{y}$  is an unconfident prediction,  $\epsilon$  should be set a large value.

## 6.5 Experiments

### 6.5.1 Experimental setup

#### 6.5.1.1 Datasets

The three datasets we chose for the experiments are broadly used in related studies. These are:

- **MNIST** [62], which consists of 70,000 handwritten digit images in 10 classes: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Each image has been resized to  $32 \times 32$ .
- **Fashion-MNIST** [32] consisting of 70,000 images across 10 classes, including T-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag and ankle boot. Again, each image has been resized to  $32 \times 32$ .
- **CIFAR10** [61] with 60,000 images across 10 classes, including airplane, automobile, bird, cat, deer, dog, horse, ship and truck, also resized to  $32 \times 32$ .

**Table 6.1** presents the data allocation in our experiments. Note that the size of the attacker’s training set is 60,000 for MNIST and Fashion-MNIST and 50,000 for CIFAR10. In the real world, it is infeasible for an attacker to collect a great many samples that

share the same distribution with the training set of a target model. In the experiments, we attempt to build a strong attacker who can collect a large number of samples. Then, successfully defeating this strong attacker can prove the effectiveness of the proposed defense method.

Dataset	$D_{\text{target}}^{\text{train}}$	$D_{\text{target}}^{\text{test}}$	$D_{\text{attack}}^{\text{train}}$	$D_{\text{attack}}^{\text{test}}$
MNIST	60,000	10,000	60,000	10,000
Fashion-MNIST	60,000	10,000	60,000	10,000
CIFAR10	50,000	10,000	50,000	10,000

Table 6.1: Data allocation

### 6.5.1.2 Target models

We used the architecture proposed in [133] for the three datasets, which consists of three CNN blocks, two fully-connected layers and a softmax function. Each CNN block consists of a convolutional layer followed by a batch normalization layer, a max-pooling layer and a ReLU activation layer. The two fully-connected layers are added after the CNN blocks. Finally, the softmax function is added to the last layer to convert arbitrary neural signals into a valid confidence score vector  $\mathbf{y}$ .

Table 6.2: Comprehensive results of the three defense methods

Dataset	Defense methods	Utility			Model inversion	Membership inference		Time overhead	
		Train acc.	Test acc.	Conf. dist.	Inversion error	ML-Leaks	NSH	Train (h)	Test (s)
MNIST	No defense	99.94%	99.63%	0	0.935	70.4%	72.3%	0	0
	<b>DP-based</b> ( $\epsilon = 0.1$ )	<b>99.94%</b>	<b>99.63%</b>	<b>0.948</b>	<b>0.924</b>	<b>49.8%</b>	<b>50.3%</b>	<b>0</b>	<b>6.96e-05</b>
	<b>DP-based</b> ( $\epsilon = 0.7$ )	<b>99.94%</b>	<b>99.63%</b>	<b>0.783</b>	<b>0.925</b>	<b>50.4%</b>	<b>50.9%</b>	<b>0</b>	<b>6.95e-05</b>
	<b>DP-based</b> ( $\epsilon = 1.4$ )	<b>99.94%</b>	<b>99.63%</b>	<b>0.535</b>	<b>0.927</b>	<b>51.1%</b>	<b>51.6%</b>	<b>0</b>	<b>6.97e-05</b>
	<b>DP-based</b> ( $\epsilon = 2.0$ )	<b>99.94%</b>	<b>99.63%</b>	<b>0.328</b>	<b>0.928</b>	<b>52.0%</b>	<b>52.5%</b>	<b>0</b>	<b>6.95e-05</b>
	MemGuard	99.94%	99.63%	0.392	0.908	57.2%	55.3%	1.02	2.53
	Purification	99.87%	99.55%	0.287	0.925	65.5%	68.2%	6.31	1.22e-04
Fashion-MNIST	No defense	99.74%	92.73%	0	0.708	69.3%	71.3%	0	0
	<b>DP-based</b> ( $\epsilon = 0.1$ )	<b>99.74%</b>	<b>92.73%</b>	<b>0.948</b>	<b>0.691</b>	<b>49.7%</b>	<b>50.1%</b>	<b>0</b>	<b>6.91e-05</b>
	<b>DP-based</b> ( $\epsilon = 0.7$ )	<b>99.74%</b>	<b>92.73%</b>	<b>0.774</b>	<b>0.694</b>	<b>50.5%</b>	<b>50.8%</b>	<b>0</b>	<b>6.91e-05</b>
	<b>DP-based</b> ( $\epsilon = 1.4$ )	<b>99.74%</b>	<b>92.73%</b>	<b>0.514</b>	<b>0.695</b>	<b>51.2%</b>	<b>51.7%</b>	<b>0</b>	<b>6.90e-05</b>
	<b>DP-based</b> ( $\epsilon = 2.0$ )	<b>99.74%</b>	<b>92.73%</b>	<b>0.316</b>	<b>0.697</b>	<b>51.9%</b>	<b>52.5%</b>	<b>0</b>	<b>6.93e-05</b>
	MemGuard	99.74%	92.73%	0.442	0.697	55.2%	54.2%	1.1	2.62
	Purification	99.68%	92.65%	0.301	0.693	65.1%	67.1%	6.40	1.22e-04
CIFAR10	No defense	99.14%	81.63%	0	0.392	65.6%	63.2%	0	0
	<b>DP-based</b> ( $\epsilon = 0.1$ )	<b>99.14%</b>	<b>81.63%</b>	<b>0.947</b>	<b>0.426</b>	<b>50.1%</b>	<b>49.3%</b>	<b>0</b>	<b>7.41e-05</b>
	<b>DP-based</b> ( $\epsilon = 0.7$ )	<b>99.14%</b>	<b>81.63%</b>	<b>0.787</b>	<b>0.428</b>	<b>50.8%</b>	<b>49.9%</b>	<b>0</b>	<b>7.40e-05</b>
	<b>DP-based</b> ( $\epsilon = 1.4$ )	<b>99.14%</b>	<b>81.63%</b>	<b>0.495</b>	<b>0.429</b>	<b>51.5%</b>	<b>50.6%</b>	<b>0</b>	<b>7.39e-05</b>
	<b>DP-based</b> ( $\epsilon = 2.0$ )	<b>99.14%</b>	<b>81.63%</b>	<b>0.305</b>	<b>0.502</b>	<b>52.3%</b>	<b>51.5%</b>	<b>0</b>	<b>7.42e-05</b>
	MemGuard	99.14%	81.63%	0.327	0.431	53.9%	52.9%	5.1	2.7
	Purification	99.09%	81.56%	0.284	0.403	61.2%	58.0%	8.24	1.24e-04

### 6.5.1.3 Attack models

We used two different attack models for the membership inference attacks.

**ML-leak attack** [98]. This is a confidence-based membership inference attack. The attacker has no knowledge of the  $D_{\text{attack}}^{\text{train}}$  and  $D_{\text{attack}}^{\text{test}}$  membership labels. Thus, a shadow model has to be trained to replicate the target model; then the attack model must be trained based on the confidence scores of the shadow model. To guarantee the strongest attack, the shadow model should have the same architecture as the target model. We use the same architecture as in [98] for the attack model which is a multi-layer perceptron with a 64-unit hidden layer and a sigmoid output layer.

**NSH attack** [79]. This is a combined confidence/label-based membership inference attack. The attacker has knowledge of the  $D_{\text{attack}}^{\text{train}}$  and  $D_{\text{attack}}^{\text{test}}$  membership labels. Thus, no shadow model is needed. The adversary can simply directly query the target model to receive the confidence score vectors. The architecture is the same as in [79] which consists of three neural networks. The first has the layers of size: [100,1024,512,64] and takes confidence score vectors as input. The second has the layers of size: [100,512,64] and takes labels as input. The third network has the layers of size: [256,64,1] and takes the outputs of the first and second networks as input.

For the model inversion attacks, we adopted the model proposed in [133].

**Adversarial model inversion attack** [133]. The adversary trains an inversion model to infer reconstruction of the input data record. We use the same inversion model architecture as in [133] which consists of four transposed CNN blocks. The first three blocks each has a transposed convolutional layer followed by a sigmoid activation function that converts neural signals into real values in [0, 1].

### 6.5.1.4 Comparison defense methods

For comparison, we chose two existing defense methods that have been experimentally proven as state-of-the-art [132]. These are the **MemGuard** method [53] for the membership inference attacks and the **Purification** method [132] for the model inversion attacks.

**MemGuard** [53]. The defense model consists of three hidden layers: [256,128,64]. It uses ReLU in the hidden layers and sigmoid in the output layer.

**Purification** [132]. The defense model used is an autoencoder with the layers of size [10,7,4,7,10]. Every hidden layer uses a ReLU activation function and batch normalization.



### 6.5.1.5 Evaluation metrics

We used five metrics to evaluate the performance and efficiency of these defense methods following the specifications outlined in [132]. A brief description of each follows.

**Classification accuracy.** This metric demonstrates the performance of target models on classification tasks. It is measured on the training set  $D_{\text{target}}^{\text{train}}$  and test set  $D_{\text{target}}^{\text{test}}$  of target models.

**Confidence score distortion.** This metric shows the utility of the perturbed confidence score vectors. As analyzed in [11], the utility is measured by computing the  $l_2$  norm of the distance between an original confidence score vector, predicted by a target model, and a perturbed confidence score vector, computed using the defense method.

**Membership inference accuracy.** This metric shows the classification accuracy of attack models in predicting the membership of input data records. It is measured on  $D_{\text{target}}^{\text{train}} - D_{\text{attack}}^{\text{train}}$ , i.e., members, and  $D_{\text{target}}^{\text{test}} - D_{\text{attack}}^{\text{test}}$ , i.e., non-members.

**Inversion error.** This metric shows the reconstruction accuracy of the attack model in reconstructing the input data records. It is measured by computing the mean squared error between the original input data record and the reconstructed data record. It is measured on datasets  $D_{\text{target}}^{\text{train}}$  and  $D_{\text{target}}^{\text{test}}$ .

**Time overhead.** This metric indicates the efficiency of the defense methods. It is measured by reporting the extra time consumed by applying defense methods. The time overhead includes both the training time of any models introduced by these defense methods and the test time when using these models.

## 6.5.2 Experimental results

### 6.5.2.1 Comparison with existing defense methods

The full results of the comparisons appear in **Table 6.2**. As shown, our method significantly reduced the membership inference accuracy with no classification accuracy loss and 0 training time. By comparison, the other defense methods had a very high training time. The test time overhead of our method, i.e., perturbing the confidence score vectors, was also much lower than the other methods. MemGuard’s test time is consumed by solving an optimization problem, while for the Purification method, it is incurred in computing a forward pass of the defense model.

After applying our method, the membership inference accuracy drops about 20% on MNIST and Fashion-MNIST datasets, and about 15% on CIFAR10 dataset. Thus, our method renders the results of a membership inference attack down to little more than a

random guess for the attacker, which is better than the other two methods. Additionally, our method yielded a larger distortion in confidence scores than the other two methods. We reason this is because those two methods treat perturbing the confidence scores as an optimizing problem. Our method is also capable of less distortion, simply by setting a larger  $\epsilon$  value. However, increase in the value of  $\epsilon$  will incur the rise of membership inference attack success rates. Thus, here is a trade-off between the confidence score distortion and the membership inference attack success rate. During the experiments, we found that confidence score distortion was not a critical factor in classification accuracy. This is due to the fact that a classifier typically selects the label with the highest score as the output. Hence, any defense method can guarantee that the classification accuracy level will be maintained, as long as there is also a guarantee that the scores in the perturbed vector will be in the same order as the original vector. Therefore, we can focus on tuning  $\epsilon$  value to reduce attack success rates.

The results for model inversion attacks are shown in **Figure 6.3, 6.4 and 6.5**. We drew three interesting findings from these experiments. First, the reconstructed images reveal the average features of one class of images. This means that the images belonging to one class have a very similar reconstructed image. The second finding is that the quality of reconstructed images depends heavily on the color and background of the original images. A grey-scale image with no background usually gives rise to a much better reconstructed version than a colorful image with a very rich background. As shown in the second row in **Figure 6.5**, even without defense, the model inversion attack method [133] cannot precisely reconstruct images in CIFAR10. Then, as shown in the third row, using our defense method can make the attack results even worse. The third finding is that inversion error is not critical to the quality of the reconstructed images. This is because the aim of model inversion attack is usually for human perception. For example, slightly rotating or adding a small amount of noise to an image has a negligible impact on human perception but may induce huge mean squared errors, i.e., inversion errors. Thus, even if a reconstructed image has a huge inversion error compared with the original one, it may still be recognizable to a person. This finding also explains the converse that some reconstructed images have very bad quality despite small inversion errors. The use of an image-specific evaluation metric, e.g., structural similarity index measure (SSIM) [129], is left to future work.

From **Figure 6.3 and 6.4**, we can see that, without any defense mechanism, the attacker can infer very accurate reconstructions of the images. However, with a defense, the inversion results become vague. Our method "averages" the images using DP making

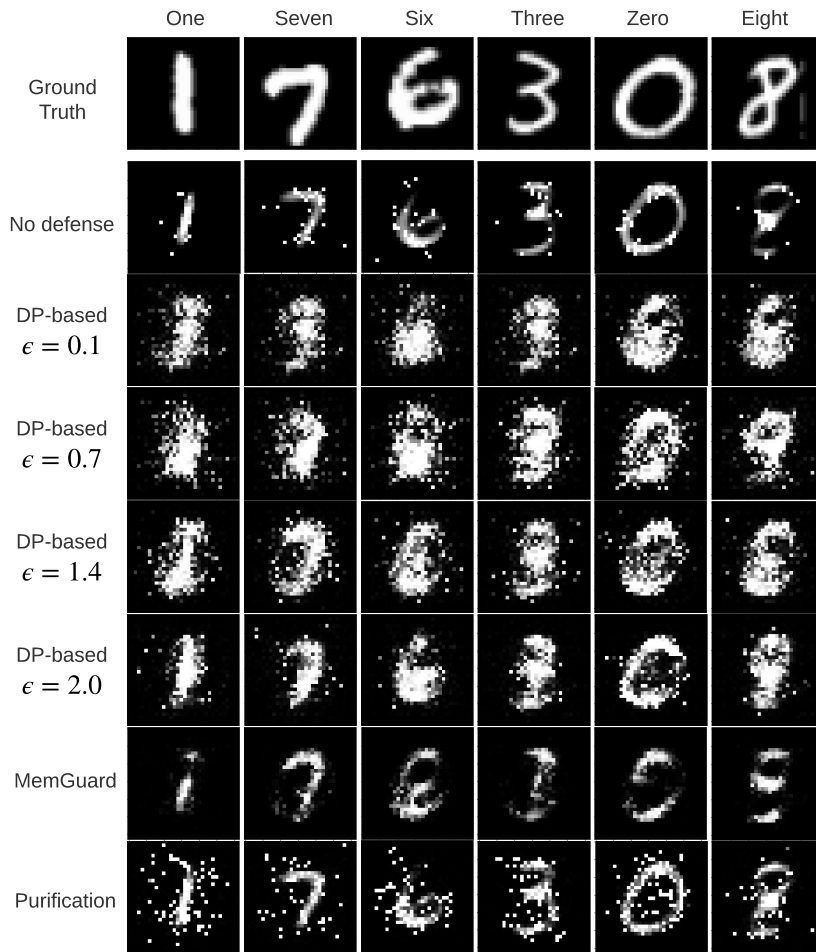


Figure 6.3: Defenses against the model inversion attacks on MNIST. Rows 3-6 show the results for our DP-based method with different  $\epsilon$  values.

them look more the same by removing useful information from the confidence score vectors. Moreover, as the  $\epsilon$  value increases, the reconstructed images become clearer. This can be explained by the fact that when  $\epsilon < \epsilon^*$ , a larger  $\epsilon$  value introduces less perturbation which implies less information removal.

Notably, MemGuard achieved very good results, even though it is not designed to defend against model inversion attacks. It also removed the "bright points" from the reconstructed images. These "bright points" represent the average features of one class. Moreover, the Purification method also did a commendable obfuscation job.

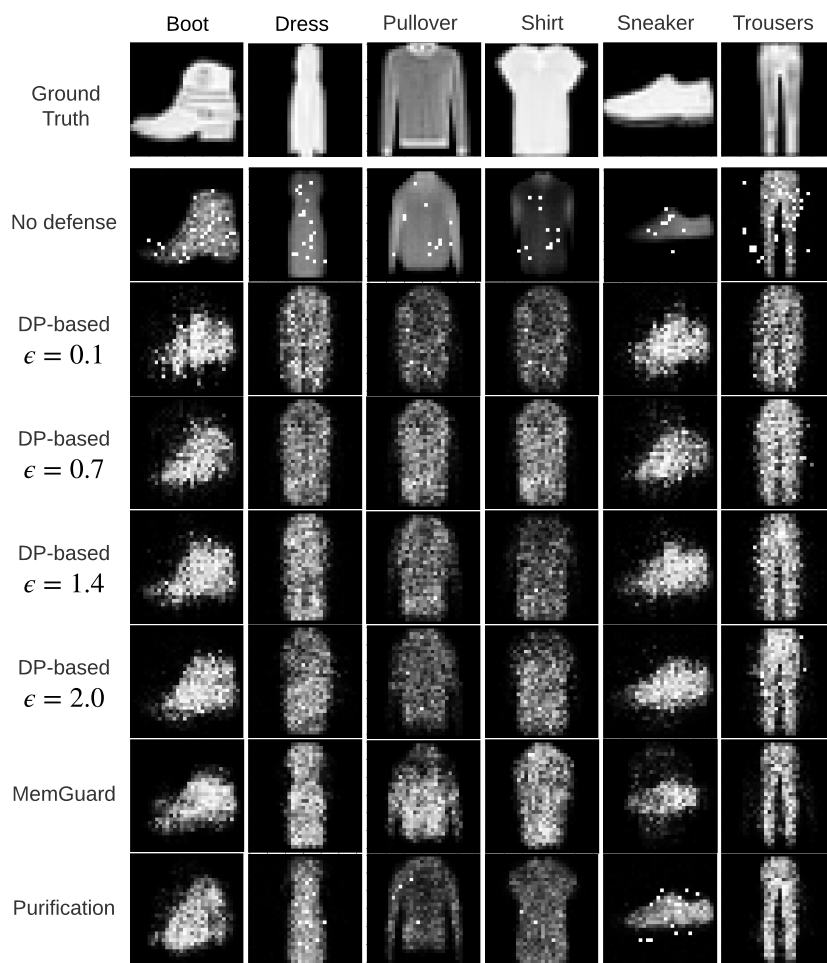


Figure 6.4: Defenses against the model inversion attacks on Fashion-MNIST. Rows 3-6 show the results for our DP-based method with different  $\epsilon$  values.

### 6.5.2.2 The impact of different $\epsilon$ values on our method

**Figures 6.6, 6.7, 6.8 and 6.9** demonstrate the impact of varying  $\epsilon$  values across the five metrics on our method. We can see that as the  $\epsilon$  value increases, classification accuracy remains the same, confidence score distortion decreases, membership inference accuracy rises, and inversion error stays mostly steady.

In terms of classification accuracy, as explained above, since our method preserves the order of the scores in a confidence vector, classification accuracy is not affected by the value of  $\epsilon$ . For confidence score distortion and membership inference accuracy, as analyzed in Section 6.4, when  $\epsilon < \epsilon^*$ , a larger  $\epsilon$  value incurs a smaller confidence score distortion which leads to higher membership inference accuracy. Hence, the confidence score distortion does not affect the classification accuracy but it does have a huge impact

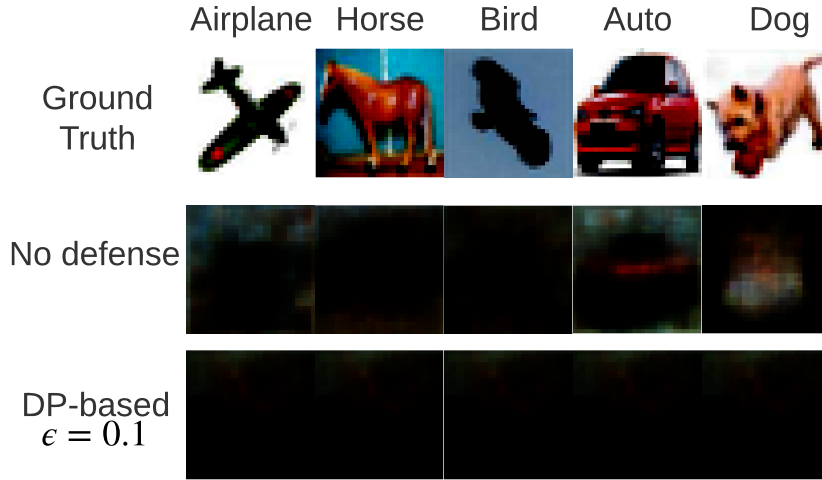
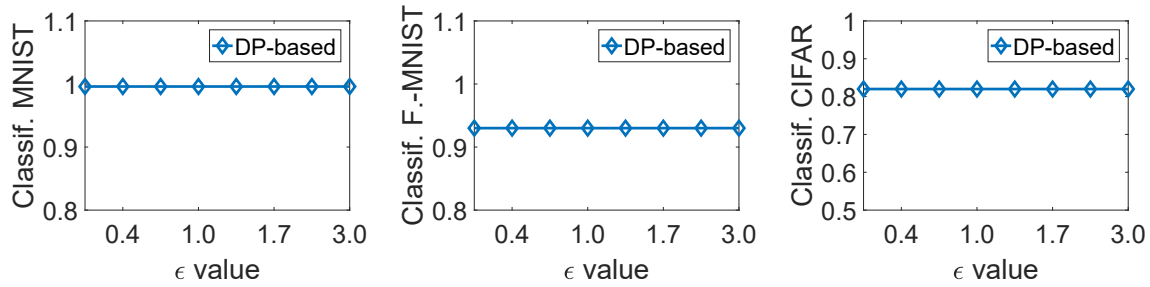


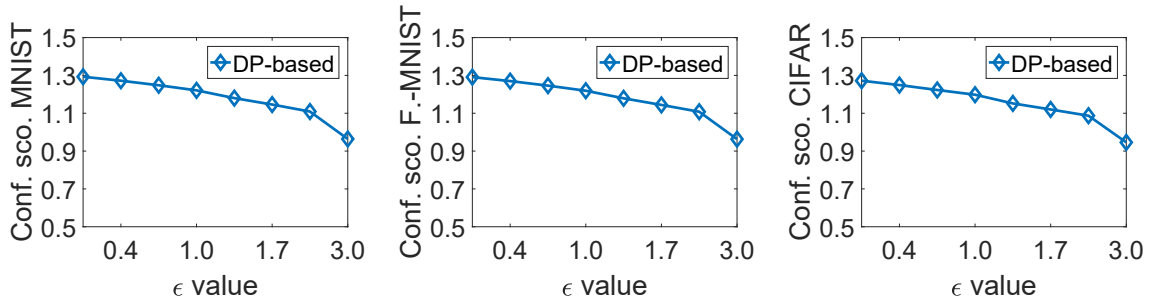
Figure 6.5: Defenses against the model inversion attacks on CIFAR10. Row 3 shows the results for our DP-based method with  $\epsilon = 0.1$ .



(a) Classification accuracy in MNIST (b) Classification accuracy in Fashion-MNIST (c) Classification accuracy in CIFAR10

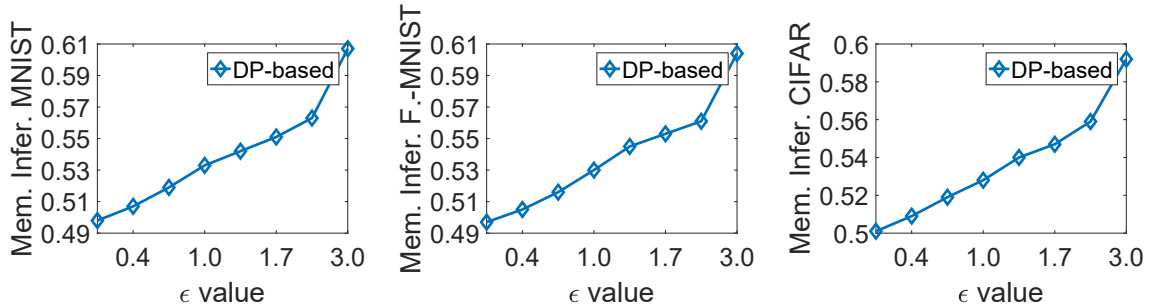
Figure 6.6: Classification accuracy with varying  $\epsilon$  values

on membership inference accuracy. Finally, the inversion error is not much affected by the value of  $\epsilon$ . This is because the inversion error is used as the loss function to train the attack model. Therefore, as long as the attack model converges, the inversion error will also converge to a relatively narrow range. It should be noted, however, that although different  $\epsilon$  values yield almost the same inversion error, they can lead to very different model inversion results, once more demonstrating that inversion errors are not a critical determinant in the model inversion results.



(a) Confidence score distortion in MNIST (b) Confidence score distortion in Fashion-MNIST (c) Confidence score distortion in CIFAR10

Figure 6.7: Confidence score distortion with varying  $\epsilon$  values



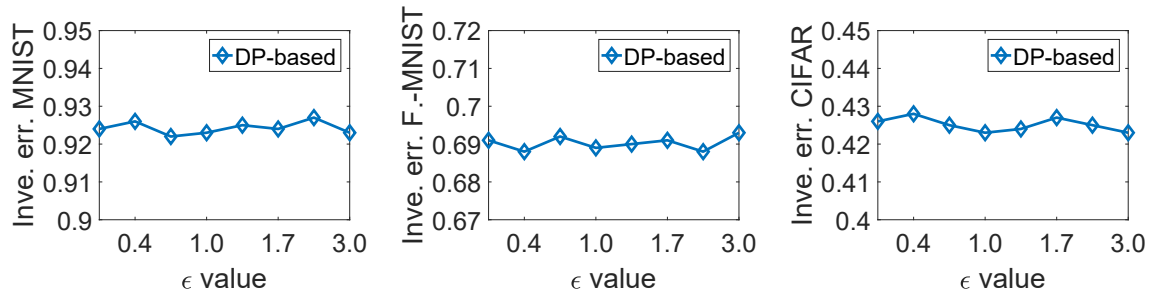
(a) Membership inference accuracy in MNIST (b) Membership inference accuracy in Fashion-MNIST (c) Membership inference accuracy in CIFAR10

Figure 6.8: Membership inference accuracy with varying  $\epsilon$  values

## 6.6 Summary and Future Work

In this chapter, we proposed a differentially private and time-efficient defense method against both membership inference attacks and model inversion attacks. Our strategy is to use an exponential mechanism to modify and normalize the confidence score vectors to confuse the attacker’s model. The experimental results show that this approach outperforms existing defense methods in various respects, especially, in terms of maintaining classification accuracy loss and not incurring training overhead.

In future, we plan to extend our method to handle the attacks that only make use of labels [19, 65]. A possible method is to modify the output label using a DP mechanism, e.g., exponential mechanism. This, certainly, will introduce a classification error. However, as every target model has an intrinsic classification error when classifying a dataset, we need only to control the introduced classification error smaller than the intrinsic classification error by properly tuning the privacy budget  $\epsilon$ . Another future work is using



(a) Inversion error in MNIST (b) Inversion error in Fashion-MNIST (c) Inversion error in CIFAR10

Figure 6.9: Inversion error with varying  $\epsilon$  values

image-specific evaluation metrics in our experiments, e.g., SSIM, to measure the quality of reconstructed images.





## CONCLUSION

This thesis addressed some privacy preservation challenges in the RL context by applying DP mechanisms. We introduced basic background theories of RL and DP in Chapter 2, and addressed variant challenges in RL and MARL systems in the following chapters. In Chapter 3, we proposed a differentially private advising framework to apply DP to allow more occurrence of advising in a MARL system. We breaks the limit of traditional advising frameworks that our method allow the occurrence of advising when two agents' actions differ in one. In Chapter 4, we proposed a differentially private planning framework for logistics-like problems in the MARL context. Our method allows multiple agents collaboratively plan the route in a private way. In Chapter 5, we applied the exponential mechanism of DP to obfuscate a DRL agent's observations, in order to protect its training policy from being inferred by privacy leakage attacks. The similar method was also applied in Chapter 6, but addressed to protect deep learning classifiers against membership inference attacks and model inversion attacks.



## BIBLIOGRAPHY

- [1] M. ABADI, A. CHU, I. GOODFELLOW, H. B. MCMAHAN, I. MIRONOV, K. TALWAR, AND L. ZHANG, *Deep learning with differential privacy*, in Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, pp. 308–318.
- [2] P. ABBEEL AND A. Y. NG, *Apprenticeship learning via inverse reinforcement learning*, in Proceedings of the twenty-first international conference on Machine learning, 2004, p. 1.
- [3] O. AMIR, E. KAMAR, A. KOLOBOV, AND B. GROSZ, *Interactive teaching strategies for agent training*, in Proceedings of IJCAI, 2016.
- [4] I. AREL, C. LIU, T. URBANIK, AND A. G. KOHLS, *Reinforcement learning-based multi-agent system for network traffic signal control*, IET Intelligent Transport Systems, 4 (2010), pp. 128–135.
- [5] S. ARORA AND P. DOSHI, *A survey of inverse reinforcement learning: Challenges, methods and progress*, Artificial Intelligence, 297 (2021), p. 103500.
- [6] K. ARULKUMARAN, M. P. DEISENROTH, M. BRUNDAGE, AND A. A. BHARATH, *Deep reinforcement learning: A brief survey*, IEEE Signal Processing Magazine, 34 (2017), pp. 26–38.
- [7] B. BALLE, M. GOMROKCHI, AND D. PRECUP, *Differentially private policy evaluation*, in International Conference on Machine Learning, PMLR, 2016, pp. 2130–2138.
- [8] A. BEIMEL, K. NISSIM, AND U. STEMMER, *Private learning and sanitization: Pure vs. approximate differential privacy*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Springer, 2013, pp. 363–378.

- [9] A. BEN-EFRAIM AND E. OMRI, *Concrete efficiency improvements for multiparty garbling with an honest majority*, in International Conference on Cryptology and Information Security in Latin America, Springer, 2017, pp. 289–308.
- [10] J. BLOCKI, A. BLUM, A. DATTA, AND O. SHEFFET, *Differentially private data analysis of social networks via restricted sensitivity*, in Proceedings of the 4th conference on Innovations in Theoretical Computer Science, 2013, pp. 87–96.
- [11] E. BOZKIR, O. GÜNLÜ, W. FUHL, R. F. SCHAEFER, AND E. KASNECI, *Differential privacy for eye tracking with temporal correlations*, Plos one, 16 (2021), p. e0255979.
- [12] R. I. BRAFMAN, *A privacy preserving algorithm for multi-agent planning and search*, in Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
- [13] R. I. BRAFMAN AND C. DOMSHLAK, *From one to many: Planning for loosely coupled multi-agent systems.*, in ICAPS, vol. 8, 2008, pp. 28–35.
- [14] C. BRAUN, K. CHATZIKOKOLAKIS, AND C. PALAMIDESSI, *Quantitative notions of leakage for one-try attacks*, Electronic Notes in Theoretical Computer Science, 249 (2009), pp. 75–91.
- [15] N. CARLINI, C. LIU, Ú. ERLINGSSON, J. KOS, AND D. SONG, *The secret sharer: Evaluating and testing unintended memorization in neural networks*, in 28th USENIX Security Symposium (USENIX Security 19), 2019, pp. 267–284.
- [16] X. CHEN, T. ZHANG, S. SHEN, T. ZHU, AND P. XIONG, *An optimized differential privacy scheme with reinforcement learning in vanet*, Computers & Security, 110 (2021), p. 102446.
- [17] H.-P. CHENG, P. YU, H. HU, F. YAN, S. LI, H. LI, AND Y. CHEN, *Leasgd: an efficient and privacy-preserving decentralized algorithm for distributed learning*, arXiv preprint arXiv:1811.11124, (2018).
- [18] Z. CHENG, D. YE, T. ZHU, W. ZHOU, P. S. YU, AND C. ZHU, *Multi-agent reinforcement learning via knowledge transfer with differentially private noise*, International Journal of Intelligent Systems, 37 (2022), pp. 799–828.

- 
- [19] C. A. CHOQUETTE-CHOO, F. TRAMER, N. CARLINI, AND N. PAPERNOT, *Label-only membership inference attacks*, in International Conference on Machine Learning, PMLR, 2021, pp. 1964–1974.
- [20] J. A. CLOUSE, *Learning from an automated training agent*, in Adaptation and learning in multiagent systems, Citeseer, 1996.
- [21] J. A. CLOUSE AND P. E. UTGOFF, *A teaching method for reinforcement learning*, in Machine Learning Proceedings 1992, Elsevier, 1992, pp. 92–101.
- [22] D. CYNTHIA, *Differential privacy*, Automata, languages and programming, (2006), pp. 1–12.
- [23] F. L. DA SILVA, R. GLATT, AND A. H. R. COSTA, *Simultaneously learning and advising in multiagent reinforcement learning*, in Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 1100–1108.
- [24] M. DE WEERDT AND B. CLEMENT, *Introduction to planning in multiagent systems*, Multiagent and Grid Systems, 5 (2009), pp. 345–355.
- [25] E. W. DIJKSTRA ET AL., *A note on two problems in connexion with graphs*, Numerische mathematik, 1 (1959), pp. 269–271.
- [26] E. H. DURFEE, C. L. ORTIZ JR, M. J. WOLVERTON, ET AL., *A survey of research in distributed, continual planning*, Ai magazine, 20 (1999), pp. 13–13.
- [27] C. DWORK, *A firm foundation for private data analysis*, Communications of the ACM, 54 (2011), pp. 86–95.
- [28] C. DWORK AND V. FELDMAN, *Privacy-preserving prediction*, in Conference On Learning Theory, PMLR, 2018, pp. 1693–1702.
- [29] C. DWORK, F. MCSHERRY, K. NISSIM, AND A. SMITH, *Calibrating noise to sensitivity in private data analysis*, in Theory of cryptography conference, Springer, 2006, pp. 265–284.
- [30] C. DWORK, A. ROTH, ET AL., *The algorithmic foundations of differential privacy.*, Found. Trends Theor. Comput. Sci., 9 (2014), pp. 211–407.

## BIBLIOGRAPHY

---

- [31] C. DWORK, A. SMITH, T. STEINKE, AND J. ULLMAN, *Exposed! a survey of attacks on private data*, Annual Review of Statistics and Its Application, 4 (2017), pp. 61–84.
- [32] FASHION-MNIST, *An MNIST-like dataset of 70,000 28x28 labeled fashion images*, in <https://www.kaggle.com/zalando-research/fashionmnist>.
- [33] C. FINN, S. LEVINE, AND P. ABBEEL, *Guided cost learning: Deep inverse optimal control via policy optimization*, in International conference on machine learning, PMLR, 2016, pp. 49–58.
- [34] C. FINN, T. YU, J. FU, P. ABBEEL, AND S. LEVINE, *Generalizing skills with semi-supervised reinforcement learning*, arXiv preprint arXiv:1612.00429, (2016).
- [35] F. FIORETTO, T. W. MAK, AND P. VAN HENTENRYCK, *Privacy-preserving obfuscation of critical infrastructure networks*, arXiv preprint arXiv:1905.09778, (2019).
- [36] F. FIORETTO, E. PONTELLI, AND W. YEOH, *Distributed constraint optimization problems and applications: A survey*, Journal of Artificial Intelligence Research, 61 (2018), pp. 623–698.
- [37] M. FREDRIKSON, S. JHA, AND T. RISTENPART, *Model inversion attacks that exploit confidence information and basic countermeasures*, in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1322–1333.
- [38] J. FU, K. LUO, AND S. LEVINE, *Learning robust rewards with adversarial inverse reinforcement learning*, arXiv preprint arXiv:1710.11248, (2017).
- [39] A. GANESH AND K. TALWAR, *Faster differentially private samplers via rényi divergence analysis of discretized langevin mcmc*, Advances in Neural Information Processing Systems, 33 (2020), pp. 7222–7233.
- [40] R. GAO AND X. MA, *Dynamic data publishing with differential privacy via reinforcement learning*, in 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), vol. 1, IEEE, 2019, pp. 746–752.
- [41] E. GARCELON, V. PERCHET, C. PIKE-BURKE, AND M. PIROTTA, *Local differential privacy for regret minimization in reinforcement learning*, Advances in Neural Information Processing Systems, 34 (2021).

- 
- [42] A. GLEAVE, M. DENNIS, C. WILD, N. KANT, S. LEVINE, AND S. RUSSELL, *Adversarial policies: Attacking deep reinforcement learning*, arXiv preprint arXiv:1905.10615, (2019).
- [43] M. GONG, Y. XIE, K. PAN, K. FENG, AND A. K. QIN, *A survey on differentially private machine learning*, IEEE computational intelligence magazine, 15 (2020), pp. 49–64.
- [44] I. J. GOODFELLOW, J. SHLENS, AND C. SZEGEDY, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572, (2014).
- [45] T. GRINSHPOUN AND T. TASSA, *A privacy-preserving algorithm for distributed constraint optimization*, in Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, 2014, pp. 909–916.
- [46] J. HAYES, L. MELIS, G. DANEZIS, AND E. DE CRISTOFARO, *Logan: evaluating privacy leakage of generative models using generative adversarial networks*, arXiv preprint arXiv:1705.07663, (2017), pp. 506–519.
- [47] M. HEIKKILÄ, E. LAGERSPETZ, S. KASKI, K. SHIMIZU, S. TARKOMA, AND A. HONKELA, *Differentially private bayesian learning on distributed data*, Advances in neural information processing systems, 30 (2017).
- [48] P. HENDERSON, R. ISLAM, P. BACHMAN, J. PINEAU, D. PRECUP, AND D. MEGER, *Deep reinforcement learning that matters*, in Proceedings of the AAAI conference on artificial intelligence, vol. 32, 2018.
- [49] J. HSU, Z. HUANG, A. ROTH, T. ROUGHGARDEN, AND Z. S. WU, *Private matchings and allocations*, SIAM Journal on Computing, 45 (2016), pp. 1953–1984.
- [50] S. HUANG, N. PAPERNOT, I. GOODFELLOW, Y. DUAN, AND P. ABBEEL, *Adversarial attacks on neural network policies*, arXiv preprint arXiv:1702.02284, (2017).
- [51] B. JAYARAMAN AND D. EVANS, *Evaluating differentially private machine learning in practice*, in 28th USENIX Security Symposium (USENIX Security 19), 2019, pp. 1895–1912.
- [52] B. JAYARAMAN, L. WANG, D. EVANS, AND Q. GU, *Distributed learning without distress: Privacy-preserving empirical risk minimization*, Advances in Neural Information Processing Systems, 31 (2018).

## BIBLIOGRAPHY

---

- [53] J. JIA, A. SALEM, M. BACKES, Y. ZHANG, AND N. Z. GONG, *Memguard: Defending against black-box membership inference attacks via adversarial examples*, in Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, 2019, pp. 259–274.
- [54] J. JORDON, J. YOON, AND M. VAN DER SCHAAR, *Pate-gan: Generating synthetic data with differential privacy guarantees*, in International conference on learning representations, 2018.
- [55] P. KAIROUZ, S. OH, AND P. VISWANATH, *Extremal mechanisms for local differential privacy*, Advances in neural information processing systems, 27 (2014).
- [56] S. P. KASIVISWANATHAN, K. NISSIM, S. RASKHODNIKOVA, AND A. SMITH, *Analyzing graphs with node differential privacy*, in Theory of Cryptography Conference, Springer, 2013, pp. 457–476.
- [57] M. KIM, O. GÜNLÜ, AND R. F. SCHAEFER, *Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication*, in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 2650–2654.
- [58] J. KOBER, J. A. BAGNELL, AND J. PETERS, *Reinforcement learning in robotics: A survey*, The International Journal of Robotics Research, 32 (2013), pp. 1238–1274.
- [59] S. KORNIENKO, O. KORNIENKO, AND J. PRIESE, *Application of multi-agent planning to the assignment problem*, Computers in Industry, 54 (2004), pp. 273–290.
- [60] J. KOS AND D. SONG, *Delving into adversarial attacks on deep policies*, arXiv preprint arXiv:1705.06452, (2017).
- [61] A. KRIZHEVSKY, V. NAIR, AND G. HINTON, *The cifar-10 dataset*, in <http://www.cs.toronto.edu/kriz/cifar.html>, 2014.
- [62] Y. LECUN, *The mnist database of handwritten digits*, in <http://yann.lecun.com/exdb/mnist/>, 1998.
- [63] J. LESKOVEC, K. J. LANG, A. DASGUPTA, AND M. W. MAHONEY, *Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters*, Internet Mathematics, 6 (2009), pp. 29–123.



- 
- [64] S. LEVINE, Z. POPOVIC, AND V. KOLTUN, *Nonlinear inverse reinforcement learning with gaussian processes*, Advances in neural information processing systems, 24 (2011).
- [65] Z. LI AND Y. ZHANG, *Membership leakage in label-only exposures*, in Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021, pp. 880–895.
- [66] T. P. LILLICRAP, J. J. HUNT, A. PRITZEL, N. HEESS, T. EREZ, Y. TASSA, D. SILVER, AND D. WIERSTRA, *Continuous control with deep reinforcement learning*, arXiv preprint arXiv:1509.02971, (2015).
- [67] Y.-C. LIN, Z.-W. HONG, Y.-H. LIAO, M.-L. SHIH, M.-Y. LIU, AND M. SUN, *Tactics of adversarial attack on deep reinforcement learning agents*, arXiv preprint arXiv:1703.06748, (2017).
- [68] W.-X. LIU, J. CAI, Y. WANG, Q. C. CHEN, AND D. TANG, *Mix-flow scheduling using deep reinforcement learning for software-defined data-center networks*, Internet Technology Letters, 2 (2019), p. e99.
- [69] R. MACLIN AND J. W. SHAVLIK, *Creating advice-taking reinforcement learners*, Machine Learning, 22 (1996), pp. 251–281.
- [70] P. MAHADEVAN, D. KRIOUKOV, K. FALL, AND A. VAHDAT, *Systematic topology analysis and generation using degree correlations*, ACM SIGCOMM Computer Communication Review, 36 (2006), pp. 135–146.
- [71] S. MAHMOUDZADEH, D. M. POWERS, AND A. ATYABI, *Uuv,Äôs hierarchical de-based motion planning in a semi dynamic underwater wireless sensor network*, IEEE transactions on cybernetics, 49 (2018), pp. 2992–3005.
- [72] S. MALIAH, G. SHANI, AND R. STERN, *Stronger privacy preserving projections for multi-agent planning*, in Twenty-Sixth International Conference on Automated Planning and Scheduling, 2016.
- [73] ———, *Collaborative privacy preserving multi-agent planning*, Autonomous agents and multi-agent systems, 31 (2017), pp. 493–530.
- [74] F. MARC, A. E. FALLAH-SEGHRUCHNI, AND I. DEGIRMENCIYAN-CARTAULT, *Coordination of complex systems based on multi-agent planning: Application to*

- the aircraft simulation domain*, in International Workshop on Programming Multi-Agent Systems, Springer, 2004, pp. 224–248.
- [75] F. MCSHERRY AND K. TALWAR, *Mechanism design via differential privacy*, in 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), IEEE, 2007, pp. 94–103.
- [76] F. S. MELO, *Convergence of q-learning: A simple proof*, Institute Of Systems and Robotics, Tech. Rep, (2001), pp. 1–4.
- [77] D. MILLER, A. SUN, M. JOHNS, H. IVE, D. SIRKIN, S. AICH, AND W. JU, *Distraction becomes engagement in automated driving*, in Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 59, SAGE Publications Sage CA: Los Angeles, CA, 2015, pp. 1676–1680.
- [78] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A. A. RUSU, J. VENESS, M. G. BELLEMARE, A. GRAVES, M. RIEDMILLER, A. K. FIDJELAND, G. OSTROVSKI, ET AL., *Human-level control through deep reinforcement learning*, nature, 518 (2015), pp. 529–533.
- [79] M. NASR, R. SHOKRI, AND A. HOUMANSADR, *Machine learning with membership privacy using adversarial regularization*, in Proceedings of the 2018 ACM SIGSAC conference on computer and communications security, 2018, pp. 634–646.
- [80] ———, *Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning*, in 2019 IEEE symposium on security and privacy (SP), IEEE, 2019, pp. 739–753.
- [81] A. Y. NG, S. J. RUSSELL, ET AL., *Algorithms for inverse reinforcement learning.*, in Icml, vol. 1, 2000, p. 2.
- [82] R. NISSIM AND R. BRAFMAN, *Distributed heuristic forward search for multi-agent planning*, Journal of Artificial Intelligence Research, 51 (2014), pp. 293–332.
- [83] C. NIU, Z. ZHENG, F. WU, S. TANG, X. GAO, AND G. CHEN, *Unlocking the value of privacy: Trading aggregate statistics over private correlated data*, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2031–2040.

- 
- [84] L. NUNES AND E. OLIVEIRA, *On learning by exchanging advice*, arXiv preprint cs/0203010, (2002).
- [85] R. S. OREIFEJ, R. AL-HADDAD, R. ZAND, R. A. ASHRAF, AND R. F. DEMARA, *Survivability modeling and resource planning for self-repairing reconfigurable device fabrics*, IEEE Transactions on Cybernetics, 48 (2017), pp. 780–792.
- [86] X. PAN, W. WANG, X. ZHANG, B. LI, J. YI, AND D. SONG, *How you act tells a lot: Privacy-leaking attack on deep reinforcement learning*, in Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019, pp. 368–376.
- [87] N. PAPERNOT, M. ABADI, U. ERLINGSSON, I. GOODFELLOW, AND K. TALWAR, *Semi-supervised knowledge transfer for deep learning from private training data*, arXiv preprint arXiv:1610.05755, (2016).
- [88] N. PAPERNOT, P. MCDANIEL, AND I. GOODFELLOW, *Transferability in machine learning: from phenomena to black-box attacks using adversarial samples*, arXiv preprint arXiv:1605.07277, (2016).
- [89] N. PAPERNOT, P. MCDANIEL, S. JHA, M. FREDRIKSON, Z. B. CELIK, AND A. SWAMI, *The limitations of deep learning in adversarial settings*, in 2016 IEEE European symposium on security and privacy (EuroS&P), IEEE, 2016, pp. 372–387.
- [90] N. PAPERNOT, S. SONG, I. MIRONOV, A. RAGHUNATHAN, K. TALWAR, AND Ú. ERLINGSSON, *Scalable private learning with pate*, arXiv preprint arXiv:1802.08908, (2018).
- [91] Z. PENG, D. CUI, J. ZUO, Q. LI, B. XU, AND W. LIN, *Random task scheduling scheme based on reinforcement learning in cloud computing*, Cluster computing, 18 (2015), pp. 1595–1607.
- [92] N. PHAN, M. VU, Y. LIU, R. JIN, D. DOU, X. WU, AND M. T. THAI, *Heterogeneous gaussian mechanism: Preserving differential privacy in deep learning with provable robustness*, arXiv preprint arXiv:1906.01444, (2019).
- [93] B. PIOT, M. GEIST, AND O. PIETQUIN, *Bridging the gap between imitation learning and inverse reinforcement learning*, IEEE transactions on neural networks and learning systems, 28 (2016), pp. 1814–1826.

- [94] D. PROSERPIO, S. GOLDBERG, AND F. MCSHERRY, *Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets*, Proceedings of the VLDB Endowment, 7 (2014), pp. 637–648.
- [95] M. A. RAHMAN, T. RAHMAN, R. LAGANIÈRE, N. MOHAMMED, AND Y. WANG, *Membership inference attack against differentially private deep learning model.*, Trans. Data Priv., 11 (2018), pp. 61–79.
- [96] S. RASKHODNIKOVA AND A. SMITH, *Differentially private analysis of graphs*, Encyclopedia of Algorithms, (2016).
- [97] D. E. RUMELHART, G. E. HINTON, AND R. J. WILLIAMS, *Learning representations by back-propagating errors*, nature, 323 (1986), pp. 533–536.
- [98] A. SALEM, Y. ZHANG, M. HUMBERT, P. BERRANG, M. FRITZ, AND M. BACKES, *ML-leaks: Model and data independent membership inference attacks and defenses on machine learning models*, arXiv preprint arXiv:1806.01246, (2018).
- [99] A. E. SALLAB, M. ABDOU, E. PEROT, AND S. YOGAMANI, *Deep reinforcement learning framework for autonomous driving*, Electronic Imaging, 2017 (2017), pp. 70–76.
- [100] S. SHALEV-SHWARTZ, S. SHAMMAH, AND A. SHASHUA, *Safe, multi-agent, reinforcement learning for autonomous driving*, arXiv preprint arXiv:1610.03295, (2016).
- [101] G. SHANI, *Advances and challenges in privacy preserving planning.*, in IJCAI, 2018, pp. 5719–5723.
- [102] S. SHEKHAR AND R. I. BRAFMAN, *Representing and planning with interacting actions and privacy*, Artificial Intelligence, 278 (2020), p. 103200.
- [103] S. SHEN, T. ZHU, D. WU, W. WANG, AND W. ZHOU, *From distributed machine learning to federated learning: In the view of data privacy and security*, Concurrency and Computation: Practice and Experience, (2020).
- [104] S. SHEN, T. ZHU, D. YE, M. YANG, T. LIAO, AND W. ZHOU, *Simultaneously advising via differential privacy in cloud servers environment*, in International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2019, pp. 550–563.

- 
- [105] R. SHOKRI AND V. SHMATIKOV, *Privacy-preserving deep learning*, in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1310–1321.
- [106] R. SHOKRI, M. STRONATI, C. SONG, AND V. SHMATIKOV, *Membership inference attacks against machine learning models*, in 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.
- [107] G. SMITH, *On the foundations of quantitative information flow*, in International Conference on Foundations of Software Science and Computational Structures, Springer, 2009, pp. 288–302.
- [108] C. SONG AND V. SHMATIKOV, *The natural auditor: How to tell if someone used your words to train their model*, arXiv preprint arXiv:1811.00513, (2018), pp. 1–15.
- [109] M. ŠTOLBA AND A. KOMENDA, *The madla planner: Multi-agent planning by combination of distributed and local heuristic search*, Artificial Intelligence, 252 (2017), pp. 175–210.
- [110] M. ŠTOLBA, J. TOŽIČKA, AND A. KOMENDA, *Secure multi-agent planning algorithms*, in ECAI 2016, IOS Press, 2016, pp. 1714–1715.
- [111] —, *Quantifying privacy leakage in multi-agent planning*, ACM Transactions on Internet Technology (TOIT), 18 (2018), pp. 1–21.
- [112] R. S. SUTTON, A. G. BARTO, ET AL., *Introduction to reinforcement learning*, vol. 135, MIT press Cambridge, 1998.
- [113] C. SZEGEDY, W. ZAREMBA, I. SUTSKEVER, J. BRUNA, D. ERHAN, I. GOODFELLOW, AND R. FERGUS, *Intriguing properties of neural networks*, arXiv preprint arXiv:1312.6199, (2013).
- [114] A. TAMAR, Y. WU, G. THOMAS, S. LEVINE, AND P. ABBEEL, *Value iteration networks*, arXiv preprint arXiv:1602.02867, (2016).
- [115] T. TASSA, T. GRINSHPOUN, AND A. YANAI, *A privacy preserving collusion secure dcop algorithm*, arXiv preprint arXiv:1905.09013, (2019).
- [116] M. E. TAYLOR, N. CARBONI, A. FACHANTIDIS, I. VLAHAVAS, AND L. TORREY, *Reinforcement learning agents providing advice in complex video games*, Connection Science, 26 (2014), pp. 45–63.

- [117] E. TODOROV, T. EREZ, AND Y. TASSA, *Mujoco: A physics engine for model-based control*, in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 5026–5033.
- [118] A. TORREÑO, E. ONAINDIA, A. KOMENDA, AND M. ŠTOLBA, *Cooperative multi-agent planning: A survey*, ACM Computing Surveys (CSUR), 50 (2017), pp. 1–32.
- [119] A. TORRENO, E. ONAINDIA, AND O. SAPENA, *Fmap: Distributed cooperative multi-agent planning*, Applied Intelligence, 41 (2014), pp. 606–626.
- [120] A. TORRENO, O. SAPENA, AND E. ONAINDIA, *Global heuristics for distributed cooperative multi-agent planning*, in Proceedings of the International Conference on Automated Planning and Scheduling, vol. 25, 2015, pp. 225–233.
- [121] L. TORREY AND M. TAYLOR, *Teaching on a budget: Agents advising agents in reinforcement learning*, in Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1053–1060.
- [122] L. TORREY, T. WALKER, J. SHAVLIK, AND R. MACLIN, *Using advice to transfer knowledge acquired in one reinforcement learning task to another*, in European Conference on Machine Learning, Springer, 2005, pp. 412–424.
- [123] J. TOŽIČKA, M. ŠTOLBA, AND A. KOMENDA, *The limits of strong privacy preserving multi-agent planning*, in Twenty-Seventh International Conference on Automated Planning and Scheduling, 2017.
- [124] S. TRUEX, L. LIU, M. E. GURSOY, L. YU, AND W. WEI, *Towards demystifying membership inference attacks*, arXiv preprint arXiv:1807.09173, (2018).
- [125] J. N. TSITSIKLIS AND B. VAN ROY, *Analysis of temporal-difference learning with function approximation*, in Advances in neural information processing systems, 1997, pp. 1075–1081.
- [126] G. VIETRI, B. BALLE, A. KRISHNAMURTHY, AND S. WU, *Private reinforcement learning with pac and regret guarantees*, in International Conference on Machine Learning, PMLR, 2020, pp. 9754–9764.

- [127] A. VIRMAUX AND K. SCAMAN, *Lipschitz regularity of deep neural networks: analysis and efficient estimation*, Advances in Neural Information Processing Systems, 31 (2018).
- [128] B. WANG AND N. HEGDE, *Privacy-preserving q-learning with functional noise in continuous spaces*, Advances in Neural Information Processing Systems, 32 (2019).
- [129] Z. WANG, A. C. BOVIK, H. R. SHEIKH, AND E. P. SIMONCELLI, *Image quality assessment: from error visibility to structural similarity*, IEEE transactions on image processing, 13 (2004), pp. 600–612.
- [130] C. J. WATKINS AND P. DAYAN, *Q-learning*, Machine learning, 8 (1992), pp. 279–292.
- [131] Q. YANG, Y. LIU, T. CHEN, AND Y. TONG, *Federated machine learning: Concept and applications*, ACM Transactions on Intelligent Systems and Technology (TIST), 10 (2019), pp. 1–19.
- [132] Z. YANG, B. SHAO, B. XUAN, E.-C. CHANG, AND F. ZHANG, *Defending model inversion and membership inference attacks via prediction purification*, arXiv preprint arXiv:2005.03915, (2020).
- [133] Z. YANG, J. ZHANG, E.-C. CHANG, AND Z. LIANG, *Neural network inversion in adversarial setting via background knowledge alignment*, in Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 225–240.
- [134] D. YE, Q. HE, Y. WANG, AND Y. YANG, *An agent-based integrated self-evolving service composition approach in networked environments*, IEEE Transactions on Services Computing, (2016).
- [135] D. YE, M. ZHANG, AND A. V. VASILAKOS, *A survey of self-organization mechanisms in multiagent systems*, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47 (2016), pp. 441–461.
- [136] D. YE, T. ZHU, S. SHEN, W. ZHOU, AND P. YU, *Differentially private multi-agent planning for logistic-like problems*, IEEE Transactions on Dependable and Secure Computing, (2020).

- [137] D. YE, T. ZHU, W. ZHOU, AND S. Y. PHILIP, *Differentially private malicious agent avoidance in multiagent advising learning*, IEEE transactions on cybernetics, 50 (2019), pp. 4214–4227.
- [138] W. YEOH AND M. YOKOO, *Distributed problem solving*, AI Magazine, 33 (2012), pp. 53–53.
- [139] Y. ZHAN, H. B. AMMAR, ET AL., *Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer*, arXiv preprint arXiv:1604.03986, (2016).
- [140] C. ZHAO, S. ZHAO, M. ZHAO, Z. CHEN, C.-Z. GAO, H. LI, AND Y.-A. TAN, *Secure multi-party computation: theory, practice and applications*, Information Sciences, 476 (2019), pp. 357–372.
- [141] L. ZHAO, Q. WANG, Q. ZOU, Y. ZHANG, AND Y. CHEN, *Privacy-preserving collaborative deep learning with unreliable participants*, IEEE Transactions on Information Forensics and Security, 15 (2019), pp. 1486–1500.
- [142] C. ZHU, H.-F. LEUNG, S. HU, AND Y. CAI, *A q-values sharing framework for multiple independent q-learners*, in Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 2324–2326.
- [143] T. ZHU, G. LI, W. ZHOU, AND S. Y. PHILIP, *Differentially private data publishing and analysis: A survey*, IEEE Transactions on Knowledge and Data Engineering, 29 (2017), pp. 1619–1638.
- [144] B. D. ZIEBART, *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*, Carnegie Mellon University, 2010.
- [145] B. D. ZIEBART, A. L. MAAS, J. A. BAGNELL, A. K. DEY, ET AL., *Maximum entropy inverse reinforcement learning.*, in Aaai, vol. 8, Chicago, IL, USA, 2008, pp. 1433–1438.
- [146] M. ZIMMER, P. VIAPPANI, AND P. WENG, *Teacher-student framework: a reinforcement learning approach*, 2014.